aws

Manual do usuário

Amazon EKS

Amazon EKS: Manual do usuário

Table of Contents

O que é o Amazon EKS?

O Amazon Elastic Kubernetes Service (Amazon EKS) é um serviço gerenciado que elimina a necessidade de instalar, operar e manter o seu próprio ambiente de gerenciamento do Kubernetes na Amazon Web Services (AWS). O Kubernetes é um sistema de código aberto que automatiza o gerenciamento, a escalação e a implantação de aplicações conteinerizadas.

Recursos do Amazon EKS

Estes são os principais atributos do Amazon EKS:

Rede e autenticação seguras

O Amazon EKS integra as workloads do Kubernetes aos serviços de rede e segurança da AWS. Ele também se integra ao AWS Identity and Access Management (IAM) para fornecer autenticação para clusters do Kubernetes.

Fácil escalação de clusters

O Amazon EKS permite que você aumente ou reduza facilmente a escala dos clusters do Kubernetes de acordo com a demanda das workloads. O Amazon EKS é compatível com escalação horizontal automática do Pod com base na CPU ou em métricas personalizadas, e com a escalação automática de clusters com base na demanda da workload total.

Experiência do Kubernetes gerenciado

Você pode fazer alterações nos clusters do Kubernetes usando o `eksctl`, o AWS Management Console, a AWS Command Line Interface (AWS CLI), a API, o `kubectl` e o Terraform.

Alta disponibilidade

O Amazon EKS oferec alta disponibilidade para seu ambiente de gerenciamento em várias zonas de disponibilidade.

Integração com serviços da AWS

O Amazon EKS se integra a outros serviços da AWS, fornecendo uma plataforma abrangente para implantação e gerenciamento de aplicações conteinerizadas. Você também pode solucionar os problemas das workloads do Kubernetes mais facilmente com várias ferramentas de observabilidade.

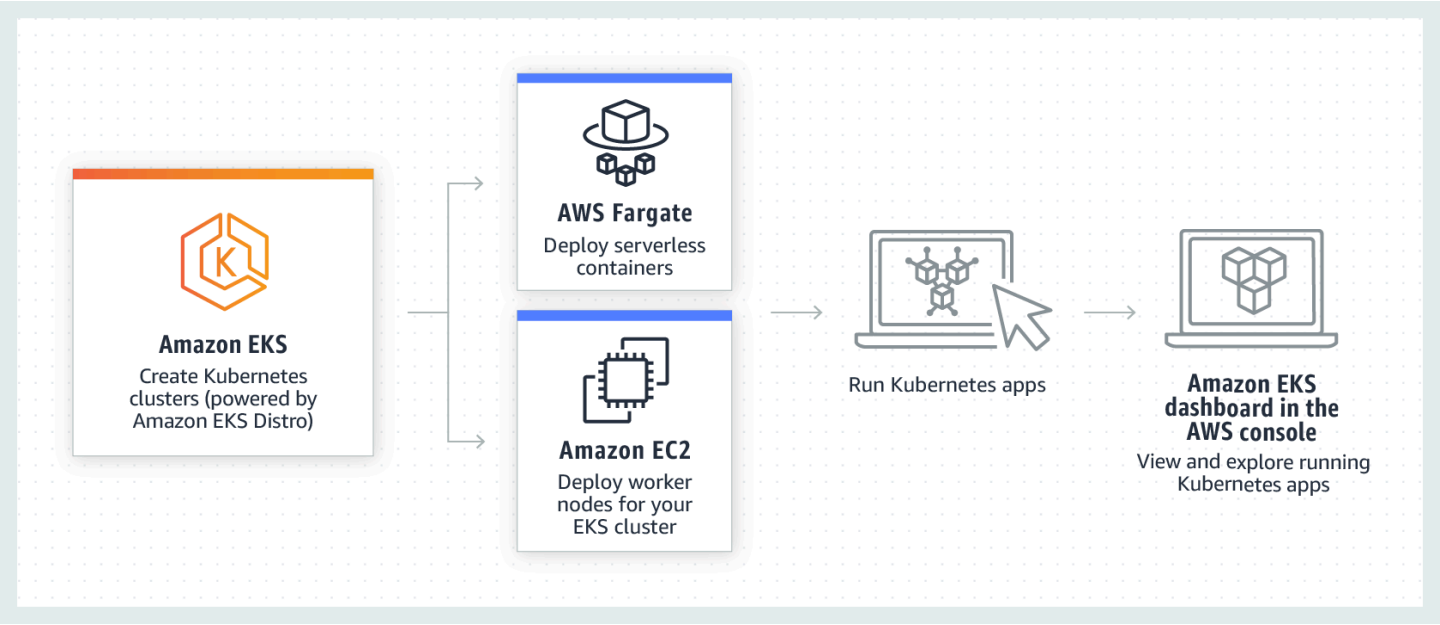
Para obter detalhes sobre todos os atributos do Amazon EKS, consulte Recursos do Amazon EKS.

Começar a usar o Amazon EKS

Para criar seu primeiro cluster e seus recursos associados, consulte Começar a usar o Amazon EKS. Em geral, começar a usar o Amazon EKS envolve as etapas a seguir.

1. Criar um cluster: comece criando o cluster usando o `eksctl`, o AWS Management Console, a AWS CLI ou um dos SDKs da AWS.
2. Escolher a abordagem dos recursos computacionais: decida entre o AWS Fargate, o Karpenter e grupos de nós gerenciados e nós autogerenciados.
3. Configuração: configure os controladores, drivers e serviços necessários.
4. Implementar workloads: ajuste as workloads do Kubernetes para utilizar da melhor maneira os recursos e a capacidade do tipo de nó escolhido.
5. Gerenciamento: supervisione as workloads, integrando os serviços do AWS para otimizar as operações e melhorar o desempenho das workloads. Você pode visualizar informações sobre as workloads usando o AWS Management Console.

O diagrama a seguir mostra um fluxo básico de execução do Amazon EKS na nuvem. Para saber mais sobre outras opções de implantação do Kubernetes, consulte Implementar clusters do Amazon EKS em ambientes locais e on-premises.



Preços do Amazon EKS

Um cluster do Amazon EKS consiste em um ambiente de gerenciamento e o Amazon Elastic Compute Cloud (Amazon EC2) ou o mecanismo de computação Fargate em que você executa os Pods. Para obter mais informações sobre a definição de preço do plano de controle, consulte Definição de preço do Amazon EKS. Tanto o Amazon EC2 quanto o Fargate fornecem:

Instâncias sob demanda

Pague pelas instâncias que você usar por segundo, sem compromisso de longo prazo nem pagamentos adiantados. Para obter mais informações, consulte Preço sob demanda do Amazon EC2 e Preço do AWS Fargate.

, Savings Plans

É possível reduzir os custos se comprometendo com uma quantidade consistente de utilização, em USD por hora, por um período de um ou de três anos. Para obter mais informações, consulte Preço e Savings Plans. Você também pode usar um modelo de preços híbrido. Por exemplo, você pode usar o Savings Plans para atender a seu tráfego regular e aumentar a escala verticalmente dos nós de clusters com instâncias spot para atender às demandas de pico.

Casos de uso comuns no Amazon EKS

O Amazon EKS oferece serviços robustos do Kubernetes gerenciados na AWS, destinados a otimizar aplicações conteinerizadas. A seguir, alguns dos casos de uso mais comuns do Amazon EKS para ajudar você a aproveitar os pontos fortes do EKS para atender as suas necessidades específicas.

Implantação de aplicações de alta disponibilidade

Usando o balanceamento de carga elástico, você pode garantir que as aplicações estejam altamente disponíveis em várias zonas de disponibilidade.

Desenvoler arquiteturas de microsserviços

Use os atributos de descoberta de serviços do Kubernetes com o AWS Cloud Map ou o Amazon VPC Lattice para desenvolver sistemas resilientes.

Automatizar o processo de lançamento de software

Gerencie os pipelines de integração contínua e implantação contínua (CICD) que simplificam o processo automatizado de desenvolvimento, teste e implantação de aplicações.

Executar aplicações de tecnologia sem servidor

Use o AWS Fargate com o Amazon EKS para executar aplicações de tecnologia sem servidor. Isso significa que você pode se concentrar exclusivamente no desenvolvimento de aplicações, enquanto o Amazon EKS e o Fargate lidam com a infraestrutura subjacente.

Executar workloads de machine learning

O Amazon EKS é compatível com estruturas populares de machine learning, como TensorFlow, MXNet e PyTorch. Graças à compatibilidade com GPU, você pode lidar até com tarefas complexas de machine learning eficazmente.

Implantar consistentemente on-premises e na nuvem

Use o Amazon EKS Anywhere para operar clusters do Kubernetes em sua própria infraestrutura usando ferramentas que são consistentes com o Amazon EKS na nuvem.

Executar processamento em lote e workloads de big data com eficiência de custos

Utilize instâncias spot para executar processamento em lote e workloads de big data, como Apache Hadoop e Spark, por uma fração do custo. Isso permite que você aproveite a capacidade não utilizada do Amazon EC2 por preços reduzidos.

Proteger a aplicação e garantir conformidade

Implemente práticas de segurança sólidas e mantenha a conformidade com o Amazon EKS, que se integra aos serviços de segurança da AWS, como o AWS Identity and Access Management (IAM), o Amazon Virtual Private Cloud (Amazon VPC) e o AWS Key Management Service (AWS KMS). Isso garante privacidade e proteção de dados conformes com os padrões do setor.

Arquitetura do Amazon EKS

O Amazon EKS está alinhado com a arquitetura geral de cluster do Kubernetes. Para obter mais informações, consulte Componentes do Kubernetes na documentação do Kubernetes. As seções a seguir resumem alguns detalhes adicionais da arquitetura do Amazon EKS.

Ambiente de gerenciamento

O Amazon EKS garante que todo cluster tenha seu próprio ambiente de gerenciamento do Kubernetes exclusivo. Esse design mantém a infraestrutura de cada cluster separada, sem sobreposições entre clusters ou contas da AWS. A configuração inclui:

Componentes distribuídos

: O ambiente de gerenciamento posiciona pelos menos duas instâncias de servidor de API e três instâncias do `etcd` em três zonas de disponibilidade da AWS dentro de uma Região da AWS.

Performance ideal

: O Amazon EKS monitora e ajusta ativamente as instâncias do ambiente de gerenciamento para manter o máximo de performance.

Resiliência

: Se uma instância do ambiente de gerenciamento falhar, o Amazon EKS a substituirá rapidamente, usando outra zona de disponibilidade se necessário.

Tempo de atividade consistente

: Executando os clusters em várias zonas de disponibilidade, um Acordo de Serviço (SLA) de disponibilidade de endpoints do servidor de API confiável é alcançado.

O Amazon EKS usa o Amazon Virtual Private Cloud (Amazon VPC) para limitar o tráfego entre os componentes do ambiente de gerenciamento dentro de um único cluster. Os componentes do ambiente de gerenciamento de um cluster não podem visualizar nem receber comunicações de outros clusters ou contas da AWS, exceto quando autorizado pelas políticas de controle de acesso baseado em perfil (RBAC) do Kubernetes.

Computação

Além do ambiente de gerenciamento, um cluster do Amazon EKS tem um conjunto de máquinas de trabalho denominadas nós. Selecionar o tipo de nó de cluster apropriado do Amazon EKS é crucial para atender aos seus requisitos específicos e otimizar a utilização dos recursos. O Amazon EKS oferece os seguintes tipos de nós primários:

AWS Fargate

Fargate é um mecanismo de computação de tecnologia sem servidor para contêineres que elimina a necessidade de gerenciar as instâncias subjacentes. Com o Fargate, você especifica as necessidades de recursos da aplicação e o AWS provisiona, escala e mantém a infraestrutura automaticamente. Essa opção é ideal para usuários que priorizam a facilidade de uso e desejam se concentrar no desenvolvimento e na implantação de aplicações, em vez de no gerenciamento da infraestrutura.

Karpenter

O Karpenter é um autoescalador de cluster do Kubernetes flexível e de alto desempenho que ajuda a melhorar a disponibilidade das aplicações e a eficiência do cluster. O Karpenter inicia a quantidade certa de recursos computacionais em resposta a alterações na carga da aplicação. Essa opção pode provisionar recursos computacionais just-in-time que atendam aos requisitos de sua workload.

Grupos de nós gerenciados

Grupos de nós gerenciados são uma combinação de automação e personalização para gerenciar um conjunto de instâncias do Amazon EC2 em um cluster do Amazon EKS. O AWS cuida de tarefas como aplicação de patches, atualização e escalação de nós, facilitando os aspectos operacionais. Em paralelo, argumentos do `kubelet` personalizados são compatíveis, ampliando as possibilidades para a aplicação de políticas avançadas de gerenciamento de CPU e memória. Além disso, eles aprimoram a segurança por meio de perfis do AWS Identity and Access Management (IAM) para contas de serviço, ao mesmo tempo que reduzem a necessidade de permissões separadas por cluster.

Nós autogerenciados

Nodos autogerenciados oferecem controle total das instâncias do Amazon EC2 dentro de um cluster do Amazon EKS. Sendo o responsável por gerenciar, escalar e manter os nós, você tem controle total sobre a infraestrutura subjacente. Essa opção é adequada para os usuários que precisam de controle granular e personalização dos nós, e estão prontos para investir tempo no gerenciamento e na manutenção da infraestrutura.

Conceitos do Kubernetes

O Amazon Elastic Kubernetes Service (Amazon EKS) é um serviço gerenciado da AWS baseado no projeto do Kubernetes de código aberto. Embora seja necessário saber como o serviço Amazon

EKS se integra à Nuvem AWS (especialmente quando você cria um cluster do Amazon EKS pela primeira vez), uma vez instalado e em execução, o cluster do Amazon EKS poderá ser usado da mesma forma que qualquer outro cluster do Kubernetes. Portanto, para começar a gerenciar clusters do Kubernetes e implantar workloads, é necessário ter pelo menos uma compreensão básica dos conceitos do Kubernetes.

Esta página divide os conceitos do Kubernetes em três seções: Por que o Kubernetes?, Clusters e Workloads. A primeira seção descreve o valor da execução de um serviço da Kubernetes, em particular como um serviço gerenciado como o Amazon EKS. A seção Workloads aborda como as aplicações do Kubernetes são criadas, armazenadas, executadas e gerenciadas. A seção Clusters apresenta os diferentes componentes que compõem os clusters do Kubernetes e quais são suas responsabilidades pela criação e manutenção dos clusters do Kubernetes.

Tópicos

À medida que você avançar nesse conteúdo, os links levarão a descrições adicionais dos conceitos do Kubernetes, tanto no Amazon EKS quanto na documentação do Kubernetes, caso deseje se aprofundar em algum dos tópicos abordados aqui. Para obter detalhes sobre como o Amazon EKS implementa o ambiente de gerenciamento e os recursos computacionais do Kubernetes, consulte Arquitetura do Amazon EKS.

Por que o Kubernetes?

O Kubernetes foi projetado para aprimorar a disponibilidade e a escalabilidade ao executar aplicações em contêineres essenciais e com qualidade de produção. Em vez de apenas executar o Kubernetes em uma única máquina (embora isso seja possível), o Kubernetes atinge esses objetivos permitindo que você execute aplicações em conjuntos de computadores que podem se expandir ou se contrair para atender à demanda. O Kubernetes inclui recursos que facilitam para você:

- Implantar aplicações em várias máquinas (usando contêineres implantados em pods)
- Monitorar a integridade de contêineres e reiniciar contêineres que falharam
- Aumentar ou reduzir a escala dos contêineres verticalmente dependendo da carga
- Atualizar os contêineres com versões novas

- Alocar recursos entre contêineres
- Equilibrar o tráfego entre máquinas

A automatização desses tipos de tarefas complexas com o Kubernetes permite que os desenvolvedores de aplicações se concentrem em criar e melhorar suas workloads de aplicações, em vez de se preocupar com a infraestrutura. O desenvolvedor normalmente cria arquivos de configuração formatados como arquivos YAML que descrevem o estado desejado da aplicação. Isso pode incluir quais contêineres executar, limites de recursos, número de réplicas de pod, alocação de CPU/memória, regras de afinidade e muito mais.

Atributos do Kubernetes

Para atingir seus objetivos, o Kubernetes oferece os seguintes atributos:

- Conteinerizado: o Kubernetes é uma ferramenta de orquestração de contêineres. Para usar o Kubernetes, suas aplicações deverão ser conteinerizadas primeiro. Dependendo do tipo de aplicação, isso pode ocorrer na forma de um conjunto de microsserviços, como trabalhos em lotes ou em outras formas. Então, suas aplicações podem utilizar um fluxo de trabalho do Kubernetes que engloba um enorme ecossistema de ferramentas e em que os contêineres podem ser armazenados como imagens em um registro de contêineres, implantados em um cluster do Kubernetes e executados em um nó disponível. É possível criar e testar contêineres individuais em seu computador local com o Docker ou outro runtime de contêiner antes de implantá-los em seu cluster do Kubernetes.
- Escalável: se a demanda por suas aplicações exceder a capacidade das instâncias em execução dessas aplicações, o Kubernetes poderá aumentar a escala verticalmente. Conforme necessário, o Kubernetes pode dizer se as aplicações precisam de mais CPU ou memória e responder expandindo automaticamente a capacidade disponível ou usando mais da capacidade existente. O aumento da escala pode ser feito no nível do pod, se houver computação suficiente disponível para executar mais instâncias da aplicação (escalação automática horizontal do pod), ou no nível do nó, se mais nós precisarem ser criados para lidar com o aumento da capacidade (Cluster Autoscaler ou Karpenter). Como a capacidade não é mais necessária, esses serviços podem excluir pods desnecessários e desligar nós desnecessários.
- Disponível: se uma aplicação ou um ou nó não estiver íntegro ou disponível, o Kubernetes poderá mover as workloads em execução para outro nó disponível. Você pode forçar o problema simplesmente excluindo uma instância em execução de uma workload ou um nó que esteja executando suas workloads. O ponto principal aqui é que as workloads poderão ser levadas para outros locais se não for mais possível executá-las onde estão.

- Declarativo: o Kubernetes usa a reconciliação ativa para verificar constantemente se o estado que você declarou para o cluster corresponde ao estado real. Ao aplicar objetos do Kubernetes a um cluster, normalmente por meio de arquivos de configuração formatados em YAML, é possível, por exemplo, solicitar a inicialização das workloads que você deseja executar no cluster. Posteriormente, é possível alterar as configurações para fazer algo como usar uma versão posterior de um contêiner ou alocar mais memória. O Kubernetes fará o que for necessário para estabelecer o estado desejado. Isso pode incluir ativar ou desativar nós, interromper e reiniciar workloads ou obter contêineres atualizados.
- Combinável: como uma aplicação geralmente consiste em vários componentes, você quer poder gerenciar um conjunto desses componentes (geralmente representado por vários contêineres) juntos. Embora o Docker Compose ofereça uma maneira de fazer isso diretamente com o Docker, o comando Kompose do Kubernetes pode ajudar você a fazer isso com o Kubernetes. Consulte Traduzir um arquivo do Docker Compose em recursos do Kubernetes para ver um exemplo de como fazer isso.
- Extensível: ao contrário de software proprietário, o projeto de código aberto do Kubernetes foi desenvolvido para ser aberto para você, estendendo o Kubernetes da maneira que você quiser para atender às suas necessidades. As APIs e os arquivos de configuração são abertos para modificações diretas. Incentivamos todos a criar seus próprios controladores para ampliar a infraestrutura e os recursos do usuário final do Kubernetes. Os webhooks permitem que você configure regras de cluster para aplicar políticas e se adaptar às mudanças nas condições. Para obter mais ideias sobre como estender clusters do Kubernetes, consulte Estender o Kubernetes.
- Portátil: muitas organizações padronizaram suas operações no Kubernetes porque isso permite que elas gerenciem todas as necessidades das respectivas aplicações da mesma forma. Os desenvolvedores podem usar os mesmos pipelines para criar e armazenar aplicações conteinerizadas. Essas aplicações podem então ser implantadas em clusters do Kubernetes executados on-premises, em nuvens, em terminais de ponto de venda em restaurantes ou em dispositivos de IoT distribuídos pelos locais remotos da empresa. Sua natureza de código aberto permite que as pessoas desenvolvam essas distribuições especiais do Kubernetes, junto com as ferramentas necessárias para gerenciá-las.

Gerenciar o Kubernetes

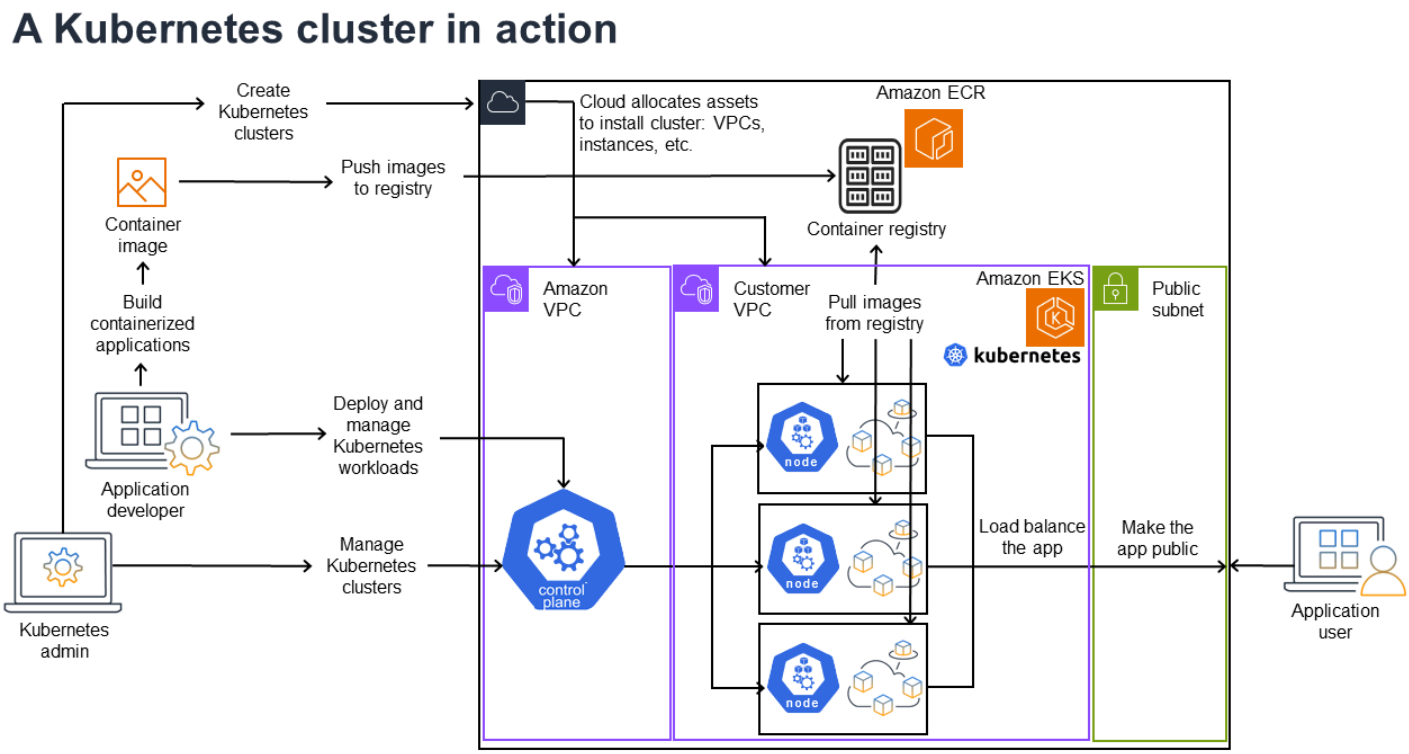
O código-fonte do Kubernetes está disponível gratuitamente, portanto, com seu próprio equipamento, você pode instalar e gerenciar o Kubernetes por conta própria. No entanto, o autogerenciamento do Kubernetes requer profundo conhecimento operacional e exige tempo e esforços para ser mantido. Por esses motivos, a maioria das pessoas que implantam workloads de produção escolhe

um provedor de nuvem (como o Amazon EKS) ou um provedor on-premises (como o Amazon EKS Anywhere) com sua própria distribuição do Kubernetes testada e o suporte de especialistas em Kubernetes. Isso permite a você descarregar grande parte do trabalho pesado indiferenciado necessário para manter seus clusters, incluindo:

- Hardware: se você não tiver hardware disponível para executar o Kubernetes de acordo com suas necessidades, um provedor de nuvem como o AWS Amazon EKS poderá ajudar a economizar nas despesas iniciais. Com o Amazon EKS, isso significa que é possível consumir os melhores recursos de nuvem oferecidos pela AWS, incluindo instâncias de computador (Amazon Elastic Compute Cloud), seu próprio ambiente privado (Amazon VPC), gerenciamento central de identidade e permissões (IAM) e armazenamento (Amazon EBS). A AWS gerencia os computadores, as redes, os data centers e todos os outros componentes físicos necessários para executar o Kubernetes. Da mesma forma, você não precisa planejar seu data center para lidar com a capacidade máxima nos dias de maior demanda. Para o Amazon EKS Anywhere ou outros clusters do Kubernetes on-premises, você é responsável por gerenciar a infraestrutura usada em suas implantações do Kubernetes, mas ainda pode contar com a AWS para manter o Kubernetes atualizado.
- Ambiente de gerenciamento: o Amazon EKS gerencia a segurança e a disponibilidade do ambiente de gerenciamento do Kubernetes hospedado pela AWS, a qual é responsável por agendar contêineres, gerenciar a disponibilidade de aplicações e outras tarefas importantes para que você possa se concentrar em suas workloads de aplicações. Se seu cluster falhar, é esperado que a AWS tenha os meios necessários para restaurá-lo para um estado de execução. Para o Amazon EKS Anywhere, você mesmo gerenciaria o ambiente de gerenciamento.
- Atualizações testadas: ao atualizar seus clusters, você pode confiar no Amazon EKS ou no Amazon EKS Anywhere para fornecer versões testadas de suas distribuições do Kubernetes.
- Complementos: existem centenas de projetos criados para ampliar e trabalhar com o Kubernetes e que podem ser adicionados à infraestrutura do seu cluster ou usar para auxiliar na execução de workloads. Para que você não precise criar e gerenciar esses complementos por conta própria, o AWS fornece complementos do Amazon EKS que você pode usar com seus clusters. O Amazon EKS Anywhere fornece pacotes selecionados que incluem compilações de muitos projetos populares de código aberto. Assim, você não precisa criar o software sozinho ou gerenciar patches críticos de segurança, correções de bugs ou atualizações. Da mesma forma, se os padrões atenderem às suas necessidades, é normal que esses complementos exijam muito pouca configuração. Consulte Estender clusters para obter detalhes sobre como estender seu cluster com complementos.

O Kubernetes em ação

O diagrama a seguir mostra as principais atividades que você realizaria como administrador ou desenvolvedor de aplicações do Kubernetes para criar e usar um cluster do Kubernetes. No processo, ele ilustra como os componentes do Kubernetes interagem entre si usando a Nuvem AWS como exemplo do provedor de nuvem subjacente.



Um administrador do Kubernetes cria o cluster do Kubernetes usando uma ferramenta específica para o tipo de provedor no qual o cluster será criado. Este exemplo usa a Nuvem AWS como provedor, que oferece o serviço de Kubernetes gerenciado chamado Amazon EKS. O serviço gerenciado aloca automaticamente os recursos necessários para criar o cluster, incluindo a criação de duas novas nuvens privadas virtuais (Amazon VPCs) para o cluster, a configuração da rede, o mapeamento de permissões do Kubernetes para gerenciar ativos na nuvem, a verificação de que os serviços do ambiente de gerenciamento têm locais para executar e a alocação de zero ou mais instâncias do Amazon EC2 como nós do Kubernetes para executar workloads. A AWS gerencia uma Amazon VPC em si para o ambiente de gerenciamento, enquanto a outra Amazon VPC contém os nós de clientes que executam workloads.

Muitas das tarefas futuras do administrador do Kubernetes serão realizadas com ferramentas do Kubernetes como o `kubectl`. Essa ferramenta faz solicitações de serviços diretamente no ambiente

de gerenciamento do cluster. As maneiras pelas quais as consultas e alterações são feitas no cluster são então muito semelhantes às formas como você as faria em qualquer cluster do Kubernetes.

Um desenvolvedor de aplicações que deseja implantar workloads nesse cluster pode realizar várias tarefas. O desenvolvedor precisa criar a aplicação em uma ou mais imagens de contêiner e, em seguida, enviar essas imagens para um registro de contêineres acessível pelo cluster do Kubernetes. A AWS oferece o Amazon Elastic Container Registry (Amazon ECR) para esse fim.

Para executar a aplicação, o desenvolvedor pode criar arquivos de configuração no formato YAML que informam ao cluster como executar a aplicação, incluindo quais contêineres extrair do registro e como agrupá-los em pods. O ambiente de gerenciamento (agendador) agenda os contêineres em um ou mais nós e o runtime do contêiner em cada nó realmente extrai e executa os contêineres necessários. O desenvolvedor também pode configurar um application load balancer para equilibrar o tráfego para os contêineres disponíveis em execução em cada nó e expor a aplicação para que ela se torne disponível em uma rede pública para o mundo externo. Com tudo isso feito, alguém que queira usar a aplicação poderá se conectar ao endpoint da aplicação para acessá-la.

A seção a seguir aborda os detalhes de cada um desses recursos, da perspectiva de clusters e workloads do Kubernetes.

Clusters

Se seu trabalho é iniciar e gerenciar clusters do Kubernetes, tenha em mente como os clusters do Kubernetes são criados, aprimorados, gerenciados e excluídos. Você também deverá saber quais são os componentes que compõem um cluster e o que precisa ser feito para mantê-los.

As ferramentas para gerenciar clusters lidam com a sobreposição entre os serviços de Kubernetes e o provedor de hardware subjacente. Por esse motivo, a automação dessas tarefas tende a ser feita pelo provedor de Kubernetes (como o Amazon EKS ou Amazon EKS Anywhere) usando ferramentas específicas para o provedor. Por exemplo, para iniciar um cluster do Amazon EKS, é possível usar `eksctl create cluster`, enquanto que para o Amazon EKS Anywhere é possível usar `eksctl anywhere create cluster`. Observe que, embora esses comandos criem um cluster do Kubernetes, eles são específicos do provedor e não fazem parte do projeto de Kubernetes em si.

Ferramentas de criação e gerenciamento de clusters

O projeto de Kubernetes oferece ferramentas para criar um cluster do Kubernetes manualmente. Portanto, se você quiser instalar o Kubernetes em uma única máquina ou executar o ambiente de gerenciamento em uma máquina e adicionar nós manualmente, poderá usar ferramentas de CLI,

como kind, minikube ou kubeadm, listadas em Ferramentas de instalação do Kubernetes. Para simplificar e automatizar todo o ciclo de vida da criação e do gerenciamento de clusters, é muito mais fácil usar ferramentas com suporte de um provedor de Kubernetes estabelecido, como o Amazon EKS ou o Amazon EKS Anywhere.

Na Nuvem AWS, é possível criar clusters do Amazon EKS usando ferramentas de CLI, como eksctl, ou ferramentas mais declarativas, como o Terraform (consulte Amazon EKS Blueprints for Terraform). Também é possível criar um cluster no Console de Gerenciamento da AWS. Consulte Recursos do Amazon EKS para ver uma lista do que o Amazon EKS oferece. As responsabilidades de Kubernetes que o Amazon EKS assume por você incluem:

- Ambiente de gerenciamento gerenciado: o AWS garante que o cluster do Amazon EKS esteja disponível e seja escalável, pois ele gerencia o ambiente de gerenciamento para você e o disponibiliza em todas as zonas de disponibilidade da AWS.
- Gerenciamento de nós: em vez de adicionar nós manualmente, você pode fazer com que o Amazon EKS crie nós automaticamente conforme necessário, seja usando grupos de nós gerenciados ou o Karpenter. Os grupos de nós gerenciados têm integrações com o escalonamento automático de clusters do Kubernetes. Usando ferramentas de gerenciamento de nós, você pode se beneficiar da economia de custos graças a recursos como instâncias spot, consolidação e disponibilidade de nós e utilização de recursos de programação para definir como as workloads são implantadas e os nós são selecionados.
- Rede de clusters: usando modelos do CloudFormation, o `eksctl` configura a rede entre os componentes do ambiente de gerenciamento e do plano de dados (nó) no cluster do Kubernetes. Ele também configura endpoints por meio dos quais as comunicações internas e externas podem ocorrer. Consulte Desmistificar redes de cluster para nós de processamento do Amazon EKS para obter detalhes. A comunicação entre pods no Amazon EKS é feita via Identidades de pods do Amazon EKS, que fornecem um meio de permitir que os pods usem métodos de gerenciamento de credenciais e permissões na Nuvem AWS.
- Complementos: o Amazon EKS evita que você precise criar e adicionar componentes de software que são comumente usados para oferecer suporte a clusters do Kubernetes. Por exemplo, quando você cria um cluster do Amazon EKS via Console de Gerenciamento da AWS, ele adiciona automaticamente o kube-proxy do Amazon EKS, o plug-in do Amazon VPC CNI para Kubernetes e os complementos do CoreDNS. Consulte Complementos do Amazon EKS para saber mais sobre esses complementos, incluindo uma lista de quais estão disponíveis.

Para executar seus clusters em seus próprios computadores e redes on-premises, a Amazon oferece o Amazon EKS Anywhere. Em vez de a Nuvem AWS atuar como o provedor, existe a opção

de executar o Amazon EKS Anywhere nas plataformas VMware vSphere, bare metal (provedor Tinkerbell), Snow, CloudStack ou Nutanix usando seu próprio equipamento.

O Amazon EKS Anywhere é baseado no mesmo software Amazon EKS Distro usado pelo Amazon EKS. No entanto, o Amazon EKS Anywhere depende de diferentes implementações da interface da API do Cluster do Kubernetes (CAPI) para gerenciar todo o ciclo de vida das máquinas em um cluster do Amazon EKS Anywhere (como CAPV para vSphere e CAPC para CloudStack). Como todo o cluster está sendo executado em seu equipamento, você assume a responsabilidade adicional por gerenciar o ambiente de gerenciamento e fazer backup de seus dados (consulte etcd posteriormente neste documento).

Componentes do cluster

Os componentes do cluster do Kubernetes são divididos em duas áreas principais: ambiente de gerenciamento e nós de processamento. Os componentes do ambiente de gerenciamento gerenciam o cluster e fornecem acesso às suas APIs. Os nós de processamento (às vezes chamados apenas de nós) fornecem os locais onde as workloads reais são executadas. Os componentes do nó consistem em serviços que são executados em cada nó para se comunicar com o plano de controle e executar contêineres. O conjunto de nós de processamento do cluster é chamado de plano de dados.

Ambiente de gerenciamento

O ambiente de gerenciamento consiste em um conjunto de serviços que gerenciam o cluster. Todos esses serviços podem ser executados em um único computador ou estar espalhados por vários computadores. Internamente, elas são chamadas de instâncias do ambiente de gerenciamento (CPIs). A forma como as CPIs são executadas depende do tamanho do cluster e dos requisitos de alta disponibilidade. Conforme a demanda aumenta no cluster, um serviço de ambiente de gerenciamento pode ser escalado para fornecer mais instâncias desse serviço, com as solicitações sendo balanceadas entre as instâncias.

As tarefas realizadas pelos componentes do ambiente de gerenciamento do Kubernetes incluem:

- Comunicação com os componentes do cluster (servidor de API): o servidor de API (kube-apiserver) expõe a API do Kubernetes para que as solicitações ao cluster possam ser feitas interna e externamente ao cluster. Em outras palavras, as solicitações para adicionar ou alterar os objetos de um cluster (pods, serviços, nós etc.) podem vir de comandos externos, como solicitações de `kubectl` para executar um pod. Da mesma forma, as solicitações podem ser feitas do servidor

da API para componentes dentro do cluster, como uma consulta ao serviço `kubelet` para saber o status de um pod.

- Armazenar dados sobre o cluster (armazenamento de valore de chave de **`etcd`**): o serviço `etcd`exerce a função crítica de acompanhar o estado atual do cluster. Se o serviço `etcd` se tornasse inacessível, você não conseguiria atualizar ou consultar o status do cluster, embora as workloads continuassem sendo executadas por algum tempo. Por esse motivo, os clusters críticos geralmente têm várias instâncias do serviço `etcd` com balanceamento de carga em execução ao mesmo tempo e fazem backups periódicos do armazenamento dos valores de chave do `etcd` para o caso de perda ou corrompimento de dados. Lembre-se de que, no Amazon EKS, tudo isso é gerenciado automaticamente para você por padrão. O Amazon EKS Anywhere fornece instruções sobre backup e restauração de etcd. Consulte o Modelo de dados do `etcd` para saber como o `etcd` gerencia dados.
- Agendar pods em nós (agendador): as solicitações para iniciar ou interromper um pod no Kubernetes são direcionadas para o Agendador do Kubernetes (kube-scheduler). Como um cluster pode ter vários nós capazes de executar o pod, cabe ao agendador escolher em qual nó (ou nós, no caso de réplicas) o pod deve ser executado. Se não houver capacidade disponível suficiente para executar o pod solicitado em um nó existente, a solicitação falhará, a menos que você tenha feito outras provisões. Essas provisões podem incluir a habilitação de serviços, como grupos de nós gerenciados ou o Karpenter, que podem iniciar automaticamente novos nós para lidar com as workloads.
- Manter os componentes no estado desejado (Controller Manager): o Kubernetes Controller Manager é executado como um processo daemon (kube-controller-manager) para observar o estado do cluster e fazer alterações no cluster para restabelecer os estados esperados. Em particular, existem vários controladores que vigiam objetos Kubernetes diferentes, incluindo um `node-lifecycle-controller`, `statefulset-controller`, `endpoint-controller`, `cronjob-controller` e outros.
- Gerenciar recursos de nuvem (Cloud Controller Manager): as interações entre Kubernetes e o provedor de nuvem que realiza as solicitações dos recursos subjacentes do data center são gerenciadas pelo Cloud Controller Manager (cloud-controller-manager). Os controladores gerenciados pelo Cloud Controller Manager podem incluir um controlador de rota (para configurar rotas de rede na nuvem), controlador de serviço (para usar serviços de balanceamento de carga na nuvem) e controlador de nós (para usar APIs de nuvem para manter os nós do Kubernetes sincronizados com os nós da nuvem).

Nós de processamento (plano de dados)

Para um cluster do Kubernetes de nó único, as workloads são executadas na mesma máquina que o ambiente de gerenciamento. No entanto, uma configuração mais normal é ter um ou mais sistemas de computador separados (nós) dedicados à execução de workloads do Kubernetes.

Quando você cria um cluster do Kubernetes pela primeira vez, algumas ferramentas de criação de clusters permitem configurar um determinado número de nós a serem adicionados ao cluster (seja identificando sistemas de computador existentes ou fazendo com que o provedor crie novos sistemas). Antes que qualquer workload seja adicionada a esses sistemas, serviços são adicionados a cada nó para implementar esses recursos:

- Gerenciar cada nó (**`kubelet`**): o servidor da API se comunica com o serviço kubelet em execução em cada nó para garantir que o nó esteja registrado corretamente e que os pods solicitados pelo agendador estejam em execução. O kubelet pode ler os manifestos do Pod e configurar volumes de armazenamento ou outros recursos necessários aos Pods no sistema local. Ele também pode verificar a integridade dos contêineres executados localmente.
- Executar contêineres em um nó (runtime de contêiner): o runtime de contêiner em cada nó gerencia os contêineres solicitados para cada pod atribuído ao nó. Isso significa que ele pode extrair imagens do contêiner do registro apropriado, executar o contêiner, interrompê-lo e responder às consultas sobre o contêiner. O runtime de contêiner padrão é containerd. A partir do Kubernetes versão 1.24, a integração especial do Docker (`dockershim`) que poderia ser usada como runtime de contêiner foi removida do Kubernetes. Embora ainda seja possível usar o Docker para testar e executar contêineres em seu sistema local, para usar o Docker com o Kubernetes, agora é necessário Instalar o mecanismo do Docker em cada nó para usá-lo com o Kubernetes.
- Gerenciar a rede entre contêineres (kube-proxy): para que seja possível oferecer suporte à comunicação entre pods usando serviços, o Kubernetes precisava de uma forma de configurar redes de pods para rastrear endereços IP e portas associadas a esses pods. O serviço kube-proxy é executado em cada um dos nós para permitir que a comunicação entre os pods ocorra.

Cluster estendido

Há alguns serviços que podem ser adicionados ao Kubernetes para oferecer suporte ao cluster, mas eles não são executados no ambiente de gerenciamento. Esses serviços geralmente são executados diretamente em nós no namespace kube-system ou em seu próprio namespace (como geralmente é feito com provedores de serviços terceirizados). Um exemplo comum é o serviço CoreDNS, que fornece serviços de DNS ao cluster. Consulte Descobrindo serviços integrados para

obter informações sobre como ver quais serviços de cluster estão sendo executados no kube-system em seu cluster.

Há diferentes tipos de complementos que podem ser adicionados aos seus clusters. Para manter seus clusters íntegros, é possível adicionar recursos de observabilidade que permitem realizar tarefas como registro em log, auditorias e métricas. Com essas informações, você pode solucionar problemas que ocorrem, geralmente por meio das mesmas interfaces de observabilidade. Exemplos desses tipos de serviço incluem o Amazon GuardDuty, CloudWatch, AWS Distro para OpenTelemetry, plug-in do Amazon VPC CNI para Kubernetes e Grafana Kubernetes Monitoring. Para armazenamento, os complementos do Amazon EKS incluem o Amazon Elastic Block Store CSI Driver (para adicionar dispositivos de armazenamento de blocos), o Amazon Elastic File System CSI Driver (para adicionar armazenamento do sistema de arquivos) e vários complementos de armazenamento de terceiros (como o driver Amazon FSx para NetApp ONTAP CSI).

Para obter uma lista mais completa dos complementos do Amazon EKS disponíveis, consulte Complementos do Amazon EKS.

Workloads

O Kubernetes define uma workload como "uma aplicação em execução no Kubernetes". Essa aplicação pode consistir em um conjunto de microsserviços executados como contêineres em pods ou pode ser executada como um trabalho em lote ou outro tipo de aplicações. O trabalho do Kubernetes é garantir que as solicitações feitas para que esses objetos sejam configurados ou implantados sejam executadas. Como alguém que implanta aplicações, você deve aprender sobre como os contêineres são criados, como os pods são definidos e quais métodos você pode usar para implantá-los.

Contêineres

O elemento mais básico de uma workload de aplicação que você implanta e gerencia no Kubernetes é um pod. Um pod representa uma forma de sustentar os componentes de uma aplicação, bem como de definir especificações que descrevem os atributos do pod. Compare isso com algo como um pacote RPM ou Deb, que agrupa software para um sistema Linux, mas não é executado como uma entidade.

Como o pod é a menor unidade implantável, ele normalmente contém um único contêiner. No entanto, vários contêineres podem existir em um pod nos casos em que os contêineres estão fortemente acoplados. Por exemplo, um contêiner de servidor Web pode ser empacotado em um pod com um tipo de contêiner auxiliar que pode fornecer registro em log, monitoramento ou outro serviço

intimamente vinculado ao contêiner do servidor Web. Nesse caso, estar no mesmo pod garante que, para cada instância em execução do pod, os dois contêineres sempre sejam executados no mesmo nó. Da mesma forma, todos os contêineres em um pod compartilham o mesmo ambiente, com os contêineres em um pod funcionando como se estivessem no mesmo host isolado. O efeito disso é que os contêineres compartilham um único endereço IP que fornece acesso ao pod e podem se comunicar entre si como se estivessem sendo executados em seu próprio host local.

As especificações do pod (`PodSpec`) definem o estado desejado do pod. É possível implantar um pod individual ou vários pods usando recursos de workload para gerenciar modelos de pod. Os recursos de workload incluem Implantações (para gerenciar várias réplicas de pod), StatefulSets (para implantar pods que precisam ser exclusivos, como pods de banco de dados) e `DaemonSets` (em que um pod precisa ser executado continuamente em cada nó). Falaremos mais sobre isso mais tarde.

Enquanto um pod é a menor unidade que você pode implantar, um contêiner é a menor unidade que você cria e gerencia.

Criar contêineres

O pod é, na verdade, apenas uma estrutura em torno de um ou mais contêineres, onde cada contêiner contém o sistema de arquivos, executáveis, arquivos de configuração, bibliotecas e outros componentes para realmente executar a aplicação. Como uma empresa chamada Docker Inc. popularizou os contêineres pela primeira vez, algumas pessoas se referem aos contêineres como contêineres Docker. No entanto, desde então, a Open Container Initiative definiu tempos de execução, imagens e métodos de distribuição de contêineres para o setor. Adicione a isso o fato de que os contêineres foram criados com base em muitos recursos existentes do Linux, outros geralmente se referem aos contêineres como contêineres OCI, contêineres Linux ou apenas contêineres.

Quando você cria um contêiner, normalmente começa com um arquivo Dockerfile (literalmente chamado assim). Dentro desse Dockerfile, é possível identificar:

- Uma imagem base: uma imagem base de contêiner é um contêiner que normalmente é construído a partir de uma versão mínima do sistema de arquivos de um sistema operacional (como Red Hat Enterprise Linux ou Ubuntu) ou de um sistema mínimo que é aprimorado para fornecer software para executar tipos específicos de aplicações (como aplicativos nodejs ou python).
- Software de aplicação: é possível adicionar o software de aplicação ao seu contêiner da mesma forma que adicionaria a um sistema Linux. Por exemplo, em seu Dockerfile, você pode executar `npm` e `yarn` para instalar um aplicativo Java ou `yum` e `dnf` para instalar pacotes RPM. Em outras

palavras, com um comando RUN em um Dockerfile, é possível executar qualquer comando que esteja disponível no sistema de arquivos da sua imagem base para instalar ou configurar o software dentro da imagem de contêiner resultante.
- Instruções: a Referência do Dockerfile descreve as instruções que podem ser adicionadas a um Dockerfile ao configurá-lo. Isso inclui instruções usadas para criar o que está no próprio contêiner (`ADD` ou `COPY` arquivos do sistema local), identificar comandos para executar quando o contêiner for executado (`CMD` ou `ENTRYPOINT`) e conectar o contêiner ao sistema em que ele é executado (identificando o `USER` de execução, um `VOLUME` local para montagem ou as portas para `EXPOSE`).

Embora o comando e o serviço `docker` tenham sido tradicionalmente usados para criar containers (`docker build`), outras ferramentas disponíveis para criar imagens de contêineres incluem `podman` e `nerdctl`. Consulte Como criar imagens de contêineres melhores ou Criar com o Docker para saber mais sobre a criação de contêineres.

Armazenamento de contêineres

Após criar sua imagem de contêiner, você poderá armazená-la em um registro de distribuição de contêineres em sua estação de trabalho ou em um registro de contêiner público. A execução de um registro de contêiner privado em sua estação de trabalho permite armazenar imagens de contêiner localmente, tornando-as prontamente disponíveis para você.

Para armazenar imagens de contêiner de forma mais pública, é possível enviá-las para um registro de contêiner público. Os registros de contêineres públicos fornecem um local central para armazenar e distribuir imagens de contêineres. Exemplos de registros de contêineres públicos incluem o Amazon Elastic Container Registry, o registro do Red Hat Quay e o registro do Docker Hub.

Ao executar workloads em contêineres no Amazon Elastic Kubernetes Service (Amazon EKS), recomendamos obter cópias de imagens oficiais do Docker armazenadas no Amazon Elastic Container Registry. AWS O Amazon ECR armazena essas imagens desde 2021. É possível pesquisar imagens de contêineres populares na Galeria Pública do Amazon ECR e, especificamente para as imagens do Docker Hub, é possível pesquisar na Galeria do Docker do Amazon ECR.

Depure contêineres em execução

Como os contêineres são criados em um formato padrão, um contêiner pode ser executado em qualquer máquina capaz de executar um runtime de contêiner (como o Docker) e cujo conteúdo corresponda à arquitetura da máquina local (como `x86_64` ou `arm`). Para testar um contêiner ou simplesmente executá-lo em seu desktop local, você pode usar nossos comandos `docker run` ou

`podman run` para iniciar um contêiner no host local. Para o Kubernetes, no entanto, cada nó de trabalho tem um runtime de contêiner implantado e cabe ao Kubernetes solicitar que um nó execute um contêiner.

Depois que um contêiner é designado para ser executado em um nó, o nó verifica se a versão solicitada da imagem do contêiner já existe nesse nó. Caso contrário, o Kubernetes solicita ao runtime do contêiner que extraia esse contêiner do registro de contêiner apropriado e, em seguida, execute esse contêiner localmente. Lembre-se de que uma imagem de contêiner se refere ao pacote de software que é movido entre seu laptop, o registro de contêiner e os nós do Kubernetes. Um contêiner se refere a uma instância em execução dessa imagem.

Pods

Quando seus contêineres estiverem prontos, trabalhar com os pods inclui configurar, implantar e tornar os pods acessíveis.

Configuração de pods

Ao definir um pod, você atribui um conjunto de atributos a ele. Esses atributos devem incluir pelo menos o nome do pod e a imagem de contêiner a ser executada. No entanto, há muitas outras coisas que podem ser configuradas com as definições do pod (consulte a página PodSpec para obter detalhes sobre o que pode ser incluído em um pod). Isso inclui:

- Armazenamento: quando um contêiner em execução for interrompido e excluído, o armazenamento de dados desse contêiner desaparecerá, a menos que você configure um armazenamento mais permanente. O Kubernetes oferece suporte a muitos tipos diferentes de armazenamento e os abstrai sob a égide de Volumes. Os tipos de armazenamento incluem CephFS, NFS, iSCSI e outros. Você pode até mesmo usar um dispositivo de blocos local conectado ao computador local. Com um desses tipos de armazenamento disponíveis em seu cluster, é possível montar o volume de armazenamento em um ponto de montagem selecionado no sistema de arquivos do seu contêiner. Um volume persistente é aquele que continua existindo depois que o pod é excluído, enquanto um volume efêmero é removido quando o pod é excluído. Se o administrador do cluster criou `StorageClasses` diferentes para seu cluster, talvez você tenha a opção de escolher os atributos do armazenamento que você usa, por exemplo, se o volume é excluído ou recuperado após o uso, se ele se expandirá se for necessário mais espaço e até mesmo se ele atende a determinados requisitos de performance.
- Segredos: ao tornar os segredos disponíveis para contêineres nas especificações do Pod, você pode fornecer as permissões de que esses contêineres precisam para acessar sistemas de arquivos, bancos de dados ou outros ativos protegidos. Chaves, senhas e tokens estão entre os

itens que podem ser armazenados como segredos. O uso de segredos faz com que você não precise armazenar essas informações em imagens de contêineres, mas apenas disponibilizá-los para contêineres em execução. `ConfigMaps` são semelhantes a segredos. Um `ConfigMap` tende a conter informações menos críticas, como pares de chave-valor para configurar um serviço.

- Recursos de contêiner: os objetos para configuração adicional de contêineres podem assumir a forma de configuração de recursos. Para cada contêiner, é possível solicitar a quantidade de memória e CPU que ele pode usar, bem como impor limites para a quantidade total desses recursos que o contêiner pode usar. Consulte Gerenciamento de recursos para pods e contêineres para obter exemplos.
- Interrupções: os pods podem ser interrompidos involuntariamente (um nó torna-se inativo) ou voluntariamente (uma atualização é necessária). Ao configurar um orçamento de interrupção do pod, é possível exercer algum controle sobre a disponibilidade da sua aplicação quando interrupções ocorrem. Consulte Especificação de um orçamento de interrupção para sua aplicação para obter exemplos.
- Namespaces: o Kubernetes fornece maneiras diferentes de isolar componentes e workloads do Kubernetes uns dos outros. Executar todos os pods de uma aplicação específica no mesmo Namespace é uma forma comum de proteger e gerenciar esses pods juntos. Você pode criar seus próprios namespaces para usar ou optar por não indicar um namespace (o que faz com que o Kubernetes use o namespace `default`). Os componentes do ambiente de gerenciamento do Kubernetes normalmente são executados no namespace `kube-system`.

A configuração descrita acima geralmente é reunida em um arquivo YAML para ser aplicada ao cluster do Kubernetes. Para clusters do Kubernetes pessoais, basta armazenar esses arquivos YAML em seu sistema local. No entanto, com clusters e workloads mais críticos, o GitOps é uma forma bastante popular de automatizar o armazenamento e as atualizações dos recursos de workload e infraestrutura do Kubernetes.

Os objetos usados para reunir e implantar as informações do pod são definidos por um dos seguintes métodos de implantação.

Implantar pods

O método que você escolheria para implantar pods depende do tipo de aplicação que você planeja executar com esses pods. Algumas das opções disponíveis são:

- Aplicações sem estado: uma aplicação sem estado não salva os dados da sessão do cliente e, assim, outra sessão não precisa fazer referência ao que aconteceu em uma sessão anterior. Isso torna mais fácil simplesmente substituir os pods por novos caso eles percam a integridade ou

movê-los sem salvar o estado. Se você estiver executando uma aplicação sem estado (como um servidor Web), poderá usar uma implantação para implantar pods e ReplicaSets. Um ReplicaSet define quantas instâncias de um pod você deseja executar simultaneamente. Embora seja possível executar um ReplicaSet diretamente, é comum executar réplicas diretamente em uma implantação para definir quantas réplicas de um pod devem ser executadas por vez.

- Aplicações com estado: uma aplicação com estado é aquela em que a identidade do pod e a ordem na qual os pods são lançados são importantes. Essas aplicações precisam de armazenamento persistente que seja estável e que precise ser implantado e escalado de forma consistente. Para implantar uma aplicação com estado em Kubernetes, é possível usar StatefulSets. Um exemplo de aplicação que normalmente é executada como StatefulSet é um banco de dados. Em um StatefulSet, é possível definir réplicas, o pod e seus contêineres, volumes de armazenamento a serem montados e locais no contêiner em que os dados são armazenados. Consulte Executar uma aplicação com estado replicada para obter um exemplo de banco de dados sendo implantado como um ReplicaSet.
- Aplicações por nó: há momentos em que você deseja executar uma aplicação em cada nó do seu cluster do Kubernetes. Por exemplo, seu data center pode exigir que cada computador execute uma aplicação de monitoramento ou um serviço de acesso remoto específico. Para o Kubernetes, é possível usar um `DaemonSet` para garantir que a aplicação selecionada seja executada em todos os nós do seu cluster.
- Aplicações executadas até a conclusão: há algumas aplicações que você deseja executar para concluir uma tarefa específica. Isso pode incluir uma que gere relatórios de status mensais ou limpe dados antigos. Um objeto Job pode ser usado para configurar uma aplicação para inicializar e executar e, em seguida, sair quando a tarefa for concluída. Um objeto CronJob permite configurar uma aplicação para ser executado em uma hora, minuto, dia do mês, mês ou dia da semana específicos usando uma estrutura definida pelo formato `crontab` do Linux.

Como tornar as aplicações acessíveis via rede

Com as aplicações frequentemente implantadas como um conjunto de microsserviços que eram movidos para lugares diferentes, o Kubernetes precisava de uma forma de fazer com que esses microsserviços se encontrassem. Além disso, para que outras pessoas pudessem acessar uma aplicação fora do cluster do Kubernetes, o Kubernetes necessitava uma forma de expor essa aplicação em endereços e portas externos. Esses recursos relacionados à rede são implementados com objetos Service e Ingress, respectivamente:

- Serviços: como um pod pode ser movido para diferentes nós e endereços, um outro pod que precisasse se comunicar com o primeiro pod poderia enfrentar dificuldades para descobrir onde

ele está. Para resolver esse problema, o Kubernetes permite representar uma aplicação como um Service. Com um Service, é possível identificar um pod ou conjunto de pods com um nome específico e, em seguida, indicar qual porta expõe o serviço dessa aplicação a partir do pod e quais portas outra aplicação poderia usar para entrar em contato com esse serviço. Outro pod em um cluster pode simplesmente solicitar um Service pelo nome, e o Kubernetes direcionará essa solicitação para a porta adequada para uma instância do pod executando esse serviço.
- Entrada: Ingress é o que pode disponibilizar aplicações representadas por serviços do Kubernetesdisponíveis para clientes que estão fora do cluster. Os recursos básicos de Ingress incluem um balanceador de carga (gerenciado pelo Ingress), o controlador do Ingress e regras para rotear solicitações do controlador para o Service. Existem vários controladores de Ingress que você pode escolher com o Kubernetes.

Próximas etapas

Compreender os conceitos básicos do Kubernetes e como eles se relacionam com o Amazon EKS ajudará você a navegar pela documentação do Amazon EKS e a documentação do Kubernetes para encontrar as informações necessárias para gerenciar clusters do Amazon EKS e implantar workloads nesses clusters. Para começar a usar o Amazon EKS, escolha uma das seguintes opções:

- Criar um cluster simples
- Criar um cluster mais complexo
- Implantar uma aplicação de exemplo
- Explorar formas de gerenciar seu cluster

Implementar clusters do Amazon EKS em ambientes locais e on-premises

Você pode usar o Amazon EKS com qualquer uma das seguintes opções de implantação:

Amazon EKS na nuvem

Você pode executar o Kubernetes na nuvem da AWS sem precisar instalar, operar e manter seu próprio ambiente de gerenciamento ou nó do Kubernetes. Essa opção é abordada neste guia.

Amazon EKS no Outposts

O AWS Outposts habilita os Serviços da AWS, a infraestrutura e os modelos operacionais nativos nas suas instalações on-premises. Com o Amazon EKS no Outposts, você pode optar por

executar clusters estendidos ou locais. Com clusters estendidos, o ambiente de gerenciamento do Kubernetes é executado em uma Região da AWS e os nós são executados no Outposts. Com clusters locais, o cluster inteiro do Kubernetes é executado localmente no Outposts, incluindo o ambiente de gerenciamento e os nós do Kubernetes. Para obter mais informações, consulte Implantar o Amazon EKS on-premises com o AWS Outposts.

Amazon EKS Anywhere

O Amazon EKS Anywhere é uma opção de implantação para o Amazon EKS que permite criar e operar facilmente clusters do Kubernetes on-premises. Tanto o Amazon EKS quanto o Amazon EKS Anywhere são baseados no Amazon EKS Distro. Para saber mais sobre o Amazon EKS Anywhere e suas diferenças do Amazon EKS, consulte Visão geral e Comparação do Amazon EKS Anywhere com o Amazon EKS na documentação do Amazon EKS Anywhere. Para obter respostas para algumas perguntas comuns, consulte Perguntas frequentes sobre o Amazon EKS Anywhere.

Amazon EKS Distro

O Amazon EKS Distro é uma distribuição do mesmo software de código aberto e das dependências do Kubernetes implantadas pelo Amazon EKS na nuvem. O Amazon EKS Distro segue o mesmo ciclo de lançamento de versões do Kubernetes que o Amazon EKS, e é fornecido como um projeto de código aberto. Para saber mais, consulte Amazon EKS Distro. Você também pode visualizar e baixar o código-fonte do Amazon EKS Distro no GitHub.

Ao escolher quais opções de implantação usar para o cluster do Kubernetes, considere o seguinte:

Atributo	Amazon EKS	Amazon EKS no Outposts	Amazon EKS Anywhere	Amazon EKS Distro
Hardware	Fornecido pela AWS	Fornecido pela AWS	Fornecido por você	Fornecido por você
Local de implantação	Nuvem da AWS	Seu data center	Seu data center	Seu data center
Local do ambiente de gerenciamento do Kubernetes	Nuvem da AWS	Nuvem AWS ou seu data center	Seu data center	Seu data center

Atributo	Amazon EKS	Amazon EKS no Outposts	Amazon EKS Anywhere	Amazon EKS Distro
Local do plano de dados do Kubernetes	Nuvem da AWS	Seu data center	Seu data center	Seu data center
Suporte	AWS Support	AWS Support	AWS Support	Suporte da comunidade do OSS

Configurar para usar o Amazon EKS

Para se preparar para o gerenciamento da linha de comando dos clusters do Amazon EKS, é necessário instalar várias ferramentas. Use o seguinte para configurar credenciais, criar e modificar clusters e trabalhar com clusters quando estiverem em execução:

- Configurar o AWS CLI: use a AWS CLI para configurar e gerenciar os serviços de que você precisa para trabalhar com os clusters do Amazon EKS. Especificamente, você precisa da AWS CLI para configurar as credenciais, mas também precisa dela com outros serviços da AWS.
- Configurar o `kubectl` e o `eksctl`: a CLI do `eksctl` interage com a AWS para criar, modificar e excluir clusters do Amazon EKS. Depois que um cluster estiver ativo, use o comando `kubectl` de código aberto para gerenciar objetos do Kubernetes nos clusters do Amazon EKS.
- Configurar um ambiente de desenvolvimento (opcional): considere adicionar as seguintes ferramentas:
  - Ferramenta de implantação local: se é a sua primeira vez usando o Kubernetes, considere instalar uma ferramenta de implantação local como `minikube`ou `kind`. Essas ferramentas permitem que você tenha um cluster do Amazon EKS em sua máquina local para testar aplicações.
  - Gerenciador de pacotes: o Helm é um gerenciador de pacotes popular do Kubernetes que simplifica a instalação e o gerenciamento de pacotes complexos. Com o Helm, fica mais fácil instalar e gerenciar pacotes como o AWS Load Balancer Controller em seu cluster do Amazon EKS.

Próximas etapas

- Configurar o AWS CLI
- Configurar o `kubectl` e o `eksctl`
- Início rápido: implantar uma aplicação Web e armazenar dados

Configurar o AWS CLI

A AWS CLI é uma ferramenta de linha de comando para trabalhar com os serviços da AWS, incluindo o Amazon EKS. Ela também é usada para autenticar usuários ou funções do IAM para acessar o cluster do Amazon EKS e outros recursos da AWS da sua máquina local. Para provisionar

recursos na AWS com base na linha de comando, você precisará obter um ID de chave de acesso da AWS e uma chave secreta para usar na linha de comando. Em seguida, você precisará configurar essas credenciais na AWS CLI. Se você ainda não instalou a AWS CLI, consulte Instalar ou atualizar para a versão mais recente da AWS CLI no AWS Command Line Interface Guia do usuário.

Para criar uma chave de acesso

1. Faça login no AWS Management Console.
2. No canto superior direito, escolha seu nome de usuário da AWS para abrir o menu de navegação. Para este exemplo, selecione **`webadmin`**. Em seguida, selecione Credenciais de segurança.
3. Em Chaves de acesso, escolha Criar chave de acesso.
4. Escolha Interface de linha de comandos (CLI) e, em seguida, escolha Próximo.
5. Selecione Create access key (Criar chave de acesso).
6. Selecione Download do arquivo .csv.

Para configurar a AWS CLI

Depois de instalar a AWS CLI, execute as etapas a seguir para configurá-la. Para obter mais informações, consulte Configurar a AWS CLI no Guia do usuário da AWS Command Line Interface.

1. Em uma janela de terminal, insira o seguinte comando:

   ```
   aws configure
   ```

   De forma alternativa, é possível configurar um perfil nomeado, como **`--profile cluster-admin`**. Se você configurar um perfil nomeado na AWS CLI, sempre deverá passar esse sinalizador nos comandos subsequentes.

2. Insira credenciais de usuário da AWS. Por exemplo:

   ```
   AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
   AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   Default region name [None]: region-code
   Default output format [None]: json
   ```

Para obter um token de segurança

Se necessário, execute o comando a seguir para obter um novo token de segurança para a AWS CLI. Para obter mais informações, consulte `get-session-token` na Referência de comandos da AWS CLI.

Por padrão, o token é válido por 15 minutos. Para alterar o tempo limite padrão da sessão, passe o sinalizador **`--duration-seconds`**. Por exemplo:

```
aws sts get-session-token --duration-seconds 3600
```

Esse comando retorna as credenciais de segurança temporárias para uma sessão da AWS CLI. Você verá a seguinte saída de resposta:

```
{
    "Credentials": {
        "AccessKeyId": "ASIA5FTRU3LOEXAMPLE",
        "SecretAccessKey": "JnKgvwfqUD9mNsPoi9IbxAYEXAMPLE",
        "SessionToken": "VERYLONGSESSIONTOKENSTRING",
        "Expiration": "2023-02-17T03:14:24+00:00"
    }
}
```

Para verificar a identidade do usuário

Se necessário, execute o comando a seguir para verificar as credenciais da AWS da sua identidade de usuário do IAM (como *`ClusterAdmin`*) para a sessão do terminal.

```
aws sts get-caller-identity
```

Esse comando retorna o nome do recurso da Amazon (ARN) da entidade do IAM que está configurada para a AWS CLI. Você verá o seguinte exemplo de saída de resposta:

```
{
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "01234567890",
    "Arn": "arn:aws:iam::01234567890:user/ClusterAdmin"
}
```

Próximas etapas

- Configurar o `kubectl` e o `eksctl`
- Início rápido: implantar uma aplicação Web e armazenar dados

Configurar o **`kubectl`** e o **`eksctl`**

`Kubectl` é uma ferramenta de linha de comando que você usa para se comunicar com o servidor de API do Kubernetes. O binário `kubectl` está disponível em muitos gerenciadores de pacotes de sistemas operacionais. Usar um gerenciador de pacotes para a instalação geralmente é mais fácil do que o processo de download e instalação manual. O comando `eksctl` permite criar e modificar clusters do Amazon EKS.

Os tópicos desta página ajudam você a instalar e configurar estas ferramentas:

- Instalar ou atualizar o `kubectl`
- Instalar o `eksctl`

Instalar ou atualizar o **`kubectl`**

Este tópico ajuda a baixar e instalar ou atualizar o binário `kubectl` no dispositivo. O binário é idêntico às versões da comunidade upstream. O binário não é exclusivo do Amazon EKS ou da AWS.

> **Note**
>
> Você deve usar uma versão do `kubectl` que esteja em uma versão secundária de diferença do plano de controle do cluster do Amazon EKS. Por exemplo, um cliente do `kubectl` `1.29` funciona com clusters do Kubernetes `1.28`, `1.29` e `1.30`.

Para instalar ou atualizar o **`kubectl`**

1. Determine se você já tem o `kubectl` instalado no dispositivo.

```
kubectl version --client
```

Se você tiver `kubectl` instalado no caminho do dispositivo, o resultado de exemplo inclui informações semelhantes às seguintes. Se quiser atualizar a versão instalada atualmente para uma versão posterior, conclua a próxima etapa, certificando-se de instalar a nova versão no mesmo local em que a versão atual está.

```
Client Version: v1.30.X-eks-1234567
```

Se você não receber nenhuma saída, você não tem o `kubectl` instalado ou ele não está instalado em um local no caminho do dispositivo.

2. Instale ou atualize o `kubectl` nos sistemas operacionais `macOS`, Linux e Windows.

   macOS

   Para instalar ou atualizar o **`kubectl`** no **`macOS`**

   1. Baixe do Amazon S3 o binário para a versão do Kubernetes do cluster.
      - Kubernetes `1.30`

        ```
        curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.2/2024-07-12/bin/darwin/amd64/kubectl
        ```

      - Kubernetes `1.29`

        ```
        curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.6/2024-07-12/bin/darwin/amd64/kubectl
        ```

      - Kubernetes `1.28`

        ```
        curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.11/2024-07-12/bin/darwin/amd64/kubectl
        ```

      - Kubernetes `1.27`

        ```
        curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.15/2024-07-12/bin/darwin/amd64/kubectl
        ```

      - Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-07-12/bin/darwin/amd64/kubectl
```

- Kubernetes `1.25`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-07-12/bin/darwin/amd64/kubectl
```

- Kubernetes `1.24`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-07-12/bin/darwin/amd64/kubectl
```

- Kubernetes `1.23`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-07-12/bin/darwin/amd64/kubectl
```

- Kubernetes `1.22`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-07-12/bin/darwin/amd64/kubectl
```

- Kubernetes `1.21`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-07-12/bin/darwin/amd64/kubectl
```

2. (Opcional) Verifique o binário baixado com a soma de verificação do `SHA-256` para o seu binário.

   a. Baixe a soma de verificação `SHA-256` para a versão Kubernetes do cluster.

   - Kubernetes `1.30`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.2/2024-07-12/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.6/2024-07-12/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.11/2024-07-12/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.15/2024-07-12/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-07-12/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.25`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-07-12/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.24`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-07-12/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.23`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-07-12/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.22`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-07-12/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.21`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-07-12/
bin/darwin/amd64/kubectl.sha256
```

b. Verifique a soma de verificação SHA-256 para o binário baixado.

```
openssl sha1 -sha256 kubectl
```

c. Verifique se a soma de verificação gerada no resultado corresponde à soma de verificação no arquivo `kubectl.sha256` baixado.

3. Aplique permissões de execução ao binário.

```
chmod +x ./kubectl
```

4. Copie o binário em uma pasta em seu PATH. Se você já tiver instalado uma versão do `kubectl`, recomendamos criar um `$HOME/bin/kubectl` e garantir que `$HOME/bin` venha primeiro em seu $PATH.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:
$PATH
```

5. (Opcional) Adicione o caminho `$HOME/bin` ao arquivo de inicialização do shell para que ele esteja configurado quando você abrir um shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bash_profile
```

Linux (amd64)

Para instalar ou atualizar o **`kubectl`** no Linux (**`amd64`**)

1. Baixe do Amazon S3 o binário do `kubectl` para a versão do Kubernetes do cluster.
   - Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.2/2024-07-12/bin/
linux/amd64/kubectl
```

   - Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.6/2024-07-12/bin/
linux/amd64/kubectl
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.11/2024-07-12/
bin/linux/amd64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.15/2024-07-12/
bin/linux/amd64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-07-12/
bin/linux/amd64/kubectl
```

- Kubernetes `1.25`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-07-12/
bin/linux/amd64/kubectl
```

- Kubernetes `1.24`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-07-12/
bin/linux/amd64/kubectl
```

- Kubernetes `1.23`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-07-12/
bin/linux/amd64/kubectl
```

- Kubernetes `1.22`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-07-12/
bin/linux/amd64/kubectl
```

- Kubernetes `1.21`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-07-12/
bin/linux/amd64/kubectl
```

2. (Opcional) Verifique o binário baixado com a soma de verificação do SHA-256 para o seu binário.

   a. Baixe a soma de verificação SHA-256 para a versão Kubernetes do cluster do Amazon S3, usando o comando para a plataforma de hardware do dispositivo.

   - Kubernetes 1.30

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.2/2024-07-12/
     bin/linux/amd64/kubectl.sha256
     ```

   - Kubernetes 1.29

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.6/2024-07-12/
     bin/linux/amd64/kubectl.sha256
     ```

   - Kubernetes 1.28

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.11/2024-07-12/
     bin/linux/amd64/kubectl.sha256
     ```

   - Kubernetes 1.27

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.15/2024-07-12/
     bin/linux/amd64/kubectl.sha256
     ```

   - Kubernetes 1.26

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-07-12/
     bin/linux/amd64/kubectl.sha256
     ```

   - Kubernetes 1.25

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-07-12/
     bin/linux/amd64/kubectl.sha256
     ```

   - Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-07-12/bin/linux/amd64/kubectl.sha256
```

- Kubernetes `1.23`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-07-12/bin/linux/amd64/kubectl.sha256
```

- Kubernetes `1.22`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-07-12/bin/linux/amd64/kubectl.sha256
```

- Kubernetes `1.21`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-07-12/bin/linux/amd64/kubectl.sha256
```

b. Verifique a soma de verificação `SHA-256` do binário baixado com um dos comandos a seguir.

- ```
  sha256sum -c kubectl.sha256
  ```

  Ao usar esse comando, confira se o seguinte comando é exibido:

  ```
  kubectl: OK
  ```

- ```
  openssl sha1 -sha256 kubectl
  ```

  Ao usar esse comando, verifique se a soma de verificação gerada no resultado corresponde à soma de verificação no arquivo `kubectl.sha256` baixado.

3. Aplique permissões de execução ao binário.

```
chmod +x ./kubectl
```

4. Copie o binário em uma pasta em seu `PATH`. Se você já tiver instalado uma versão do `kubectl`, recomendamos criar um `$HOME/bin/kubectl` e garantir que `$HOME/bin` venha primeiro em seu `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:
$PATH
```

5. (Opcional) Adicione o caminho `$HOME/bin` ao arquivo de inicialização do shell para que ele esteja configurado quando você abrir um shell.

> **Note**
>
> Essa etapa pressupõe que você esteja usando o shell Bash. Se você estiver usando outro shell, altere o comando para usar o arquivo de inicialização do shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Linux (arm64)

Para instalar ou atualizar o **`kubectl`** no Linux (**`arm64`**)

1. Baixe do Amazon S3 o binário do `kubectl` para a versão do Kubernetes do cluster.
   - Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.2/2024-07-12/bin/
linux/arm64/kubectl
```

   - Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.6/2024-07-12/bin/
linux/arm64/kubectl
```

   - Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.11/2024-07-12/
bin/linux/arm64/kubectl
```

   - Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.15/2024-07-12/
bin/linux/arm64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-07-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.25`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-07-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.24`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-07-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.23`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-07-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.22`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-07-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.21`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-07-12/bin/linux/arm64/kubectl
```

2. (Opcional) Verifique o binário baixado com a soma de verificação do `SHA-256` para o seu binário.

   a. Baixe a soma de verificação `SHA-256` para a versão Kubernetes do cluster do Amazon S3, usando o comando para a plataforma de hardware do dispositivo.

   - Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.2/2024-07-12/bin/linux/arm64/kubectl.sha256
```

   - Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.6/2024-07-12/
bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.11/2024-07-12/
bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.15/2024-07-12/
bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-07-12/
bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.25`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-07-12/
bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.24`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-07-12/
bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.23`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-07-12/
bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.22`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-07-12/
bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.21`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-07-12/bin/linux/arm64/kubectl.sha256
```

b. Verifique a soma de verificação SHA-256 do binário baixado com um dos comandos a seguir.

- ```
  sha256sum -c kubectl.sha256
  ```

  Ao usar esse comando, confira se o seguinte comando é exibido:

  ```
  kubectl: OK
  ```

- ```
  openssl sha1 -sha256 kubectl
  ```

  Ao usar esse comando, verifique se a soma de verificação gerada no resultado corresponde à soma de verificação no arquivo `kubectl.sha256` baixado.

3. Aplique permissões de execução ao binário.

```
chmod +x ./kubectl
```

4. Copie o binário em uma pasta em seu PATH. Se você já tiver instalado uma versão do `kubectl`, recomendamos criar um `$HOME/bin/kubectl` e garantir que `$HOME/bin` venha primeiro em seu `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Opcional) Adicione o caminho `$HOME/bin` ao arquivo de inicialização do shell para que ele esteja configurado quando você abrir um shell.

> **Note**
>
> Essa etapa pressupõe que você esteja usando o shell Bash. Se você estiver usando outro shell, altere o comando para usar o arquivo de inicialização do shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Windows

Para instalar ou atualizar o **`kubectl`** no Windows

1. Abra um terminal do PowerShell.
2. Baixe do Amazon S3 o binário do `kubectl` para a versão do Kubernetes do cluster.
   - Kubernetes `1.30`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.2/2024-07-12/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.29`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.6/2024-07-12/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.28`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.11/2024-07-12/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.27`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.15/2024-07-12/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.26`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-07-12/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.25`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-07-12/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.24`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.24.17/2024-07-12/bin/windows/amd64/kubectl.exe
```

- Kubernetes `1.23`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.23.17/2024-07-12/bin/windows/amd64/kubectl.exe
```

- Kubernetes `1.22`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.22.17/2024-07-12/bin/windows/amd64/kubectl.exe
```

- Kubernetes `1.21`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.21.14/2024-07-12/bin/windows/amd64/kubectl.exe
```

3. (Opcional) Verifique o binário baixado com a soma de verificação do `SHA-256` para o seu binário.

   a. Baixe a soma de verificação `SHA-256` para a versão Kubernetes para Windows do cluster.

   - Kubernetes `1.30`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.30.2/2024-07-12/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.29`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.29.6/2024-07-12/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.28`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.28.11/2024-07-12/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.27`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.27.15/2024-07-12/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.26`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.26.15/2024-07-12/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.25`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.25.16/2024-07-12/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.24`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.24.17/2024-07-12/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.23`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.23.17/2024-07-12/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.22`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.22.17/2024-07-12/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.21`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.21.14/2024-07-12/bin/windows/amd64/kubectl.exe.sha256
```

b. Verifique a soma de verificação SHA-256 para o binário baixado.

```
Get-FileHash kubectl.exe
```

c. Verifique se a soma de verificação gerada no resultado corresponde à soma de verificação no arquivo `kubectl.sha256` baixado. A saída do PowerShell deve ser uma sequência de caracteres equivalente em maiúsculas.

4. Copie o binário em uma pasta em seu PATH. Se você tiver um diretório existente no PATH que é usado para utilitários de linha de comando, copie o binário para esse diretório. Caso contrário, execute as etapas a seguir.
   a. Crie um novo diretório para os binários de linha de comando, como `C:\bin`.
   b. Copie o binário `kubectl.exe` no seu novo diretório.
   c. Edite a variável de ambiente PATH do sistema ou usuário para adicionar o novo diretório ao PATH.
   d. Feche o terminal do PowerShell e abra um novo para escolher a nova variável PATH.
3. Depois de instalar `kubectl`, você pode verificar a versão.

```
kubectl version --client
```

Quando o `kubectl` é instalado pela primeira vez, ele ainda não está configurado para se comunicar com nenhum servidor. Abordaremos essa configuração conforme necessário em outros procedimentos. Se você precisar atualizar a configuração para se comunicar com um determinado cluster específico, poderá executar o comando a seguir. Substitua *`region-code`* pela Região da AWS em que está o cluster. Substitua o *`my-cluster`* pelo nome do cluster.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Instalar o **`eksctl`**

A CLI do `eksctl` é usada para trabalhar com clusters do EKS. Ela automatiza muitas tarefas individuais. Consulte Instalação na documentação do `eksctl` para obter instruções sobre instalação do `eksctl`.

Ao usar o `eksctl`, a entidade principal de segurança do IAM que você está usando deve ter permissões para trabalhar com perfis do IAM do Amazon EKS, funções vinculadas ao serviço, o AWS CloudFormation, uma VPC e recursos relacionados. Para obter mais informações, consulte Ações, recursos e chaves de condição do Amazon Elastic Container Service for Kubernetes e Uso de funções vinculadas a serviço no Guia do usuário do IAM. Você deve concluir todas as etapas deste manual como o mesmo usuário. Execute o seguinte comando para verificar o usuário atual:

```
aws sts get-caller-identity
```

Próximas etapas

- Início rápido: implantar uma aplicação Web e armazenar dados

Início rápido: implantar uma aplicação Web e armazenar dados

Este tutorial de início rápido mostra as etapas para implantar a aplicação de amostra do jogo 2048 e persistir seus dados em um cluster do Amazon EKS usando o eksctl. O eksctl é um utilitário de infraestrutura como código que utiliza o AWS CloudFormation, o que permite que você configure um cluster totalmente funcional e completo com todos os componentes essenciais. Esses componentes incluem uma Amazon VPC e um perfil do IAM personalizados para fornecer permissões aos serviços da AWS que definimos. À medida que avançamos, orientaremos você no processo de configuração do cluster, incorporando Complementos do Amazon EKS do Amazon EKS para potencializar seu cluster com recursos operacionais. Por fim, você implantará uma amostra de workload com as anotações personalizadas necessárias para a integração total com os serviços da AWS.

Neste tutorial

Usando o modelo de cluster do eksctl a seguir, você criará um cluster do Amazon EKS com grupos de nós gerenciados. Ele configura os seguintes componentes:

Configuração de VPC

> Ao usar o modelo de cluster do eksctl a seguir, o eksctl cria automaticamente uma nuvem privada virtual (VPC) IPv4 para o cluster. Por padrão, o eksctl configura uma VPC que atende a todos os requisitos de rede, além de criar endpoints públicos e privados.

Tipo de instância

> Utilize o tipo de instância t3.medium. Esse tipo de instância oferece recursos equilibrados de computação, memória e rede, ideal para aplicações com uso moderado de CPU que podem experimentar picos ocasionais de demanda.

Autenticação

> Estabeleça os mapeamentos de IRSA para facilitar a comunicação entre os pods do Kubernetes e serviços da AWS. O modelo é configurado para definir um endpoint OpenID Connect (OIDC) para autenticação e autorização. Ele também estabelece uma conta de serviço para o AWS Load Balancer Controller (LBC), um controlador responsável por expor aplicações e gerenciar tráfego.

Persistência de dados

Integre o complemento gerenciado do driver CSI do AWS EBS para garantir a persistência dos dados da aplicação, mesmo em cenários que envolvam reinicializações ou falhas de pods. O modelo está configurado para instalar o complemento e estabelecer uma conta de serviço.

Acesso externo à aplicação

Configure e integre com o complemento do AWS Load Balancer Controller (LBC) para expor a aplicação do jogo 2048, usando o LBC para provisionar dinamicamente um Application Load Balancer (ALB).

Pré-requisitos

- Configurar para usar o Amazon EKS
- Instalar o Helm

Etapa 1: configurar o cluster

Nesta seção, você criará um cluster gerenciado baseado em grupos de nós usando instâncias t3.medium contendo dois nós. A configuração inclui uma conta de serviço para o complemento do AWS Load Balancer Controller (LBC) e a instalação da versão mais recente do driver CSI do AWS Amazon EBS. Para todos os complementos disponíveis do `eksctl`, consulte Discovering addons na documentação do `eksctl`.

- Crie um arquivo `cluster-config.yaml` e cole o conteúdo a seguir nele. Substitua region-code por uma região válida, como `us-east-1`

  O exemplo de saída é o seguinte:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: web-quickstart
  region: region-code

managedNodeGroups:
  - name: eks-mng
```

```
    instanceType: t3.medium
    desiredCapacity: 2

iam:
  withOIDC: true
  serviceAccounts:
  - metadata:
      name: aws-load-balancer-controller
      namespace: kube-system
    wellKnownPolicies:
      awsLoadBalancerController: true

addons:
  - name: aws-ebs-csi-driver
    wellKnownPolicies: # Adds an IAM service account
      ebsCSIController: true

cloudWatch:
 clusterLogging:
   enableTypes: ["*"]
   logRetentionInDays: 30
```

Etapa 2: criar um cluster

Agora, estamos prontos para criar nosso cluster do Amazon EKS. Esse processo leva vários minutos para ser concluído. Se você quiser monitorar o status, confira o console do AWS CloudFormation.

- Crie o cluster do Amazon EKS e especifique o cluster-config.yaml.

```
eksctl create cluster -f cluster-config.yaml
```

> **Note**
>
> Caso receba um `Error: checking STS access` na resposta, verifique se você está usando a identidade do usuário correta para a sessão atual do shell. Talvez você também precise especificar um perfil nomeado (por exemplo, `--profile clusteradmin`) ou obter um novo token de segurança para a AWS CLI.

Após a conclusão, você verá a seguinte saída de resposta:

```
2024-07-04 21:47:53 [#]  EKS cluster "web-quickstart" in "region-code" region is
 ready
```

Etapa 3: configurar o acesso externo às aplicações usando o AWS Load Balancer Controller (LBC)

Com o cluster operacional, nossa próxima etapa é tornar as aplicações em contêineres acessíveis externamente. Essa ação é feita por meio da implantação de um Application Load Balancer (ALB) para direcionar o tráfego fora do cluster para nossos serviços, em outras palavras, nossas aplicações. Quando criamos o cluster, estabelecemos um IAM Roles for Service Accounts (IRSA) para o Load Balancer Controller (LBC) com as permissões necessárias para criar ALBs dinamicamente, facilitando o roteamento de tráfego externo para os serviços do Kubernetes. Nesta seção, vamos configurar o AWS LBC no cluster.

Para configurar variáveis de ambiente

1. Defina a variável de ambiente `CLUSTER_REGION` para o cluster do Amazon EKS. Substitua o valor da amostra por *`region-code`*.

   ```
   export CLUSTER_REGION=region-code
   ```

2. Defina a variável de ambiente `CLUSTER_VPC` para o cluster do Amazon EKS.

   ```
   export CLUSTER_VPC=$(aws eks describe-cluster --name web-quickstart --region
    $CLUSTER_REGION --query "cluster.resourcesVpcConfig.vpcId" --output text)
   ```

Para instalar o AWS Load Balancer Controller (LBC)

O AWS Load Balancer Controller (LBC) utiliza definições de recursos personalizados (CRDs) no Kubernetes para gerenciar AWS Elastic Load Balancers (ELBs). Essas CRDs definem recursos personalizados, como balanceadores de carga e TargetGroupBindings, permitindo que o cluster do Kubernetes os reconheça e gerencie.

1. Use o Helm para adicionar o repositório de gráficos do Amazon EKS ao Helm.

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Atualize os repositórios para garantir que o Helm esteja ciente das versões mais recentes dos gráficos:

```
helm repo update eks
```

3. Execute o comando do Helm a seguir para instalar simultaneamente as definições de recursos personalizados (CRDs) e o controlador principal do AWS Load Balancer Controller (AWS LBC). Para pular a instalação das CRDs, passe o sinalizador `--skip-crds`, que poderá ser útil se as CRDs já estiverem instaladas, se for necessária uma compatibilidade de versão específica ou em ambientes com necessidades rigorosas de controle de acesso e personalização.

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
    --namespace kube-system \
    --set clusterName=web-quickstart \
    --set serviceAccount.create=false \
    --set region=${CLUSTER_REGION} \
    --set vpcId=${CLUSTER_VPC} \
    --set serviceAccount.name=aws-load-balancer-controller
```

Você verá a seguinte saída de resposta:

```
NAME: aws-load-balancer-controller
LAST DEPLOYED: Wed July 3 19:43:12 2024
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```

Etapa 4: implantar a aplicação de amostra do jogo 2048

Agora que o balanceador de carga está configurado, é hora de habilitar o acesso externo para aplicações em contêineres no cluster. Nesta seção, mostraremos as etapas para implantar o popular “jogo 2048” como uma aplicação de amostra no cluster. O manifesto fornecido inclui anotações personalizadas para o Application Load Balancer (ALB), especificamente a anotação de “scheme”

e a anotação de “target-type”. Essas anotações se integram e instruem o AWS Load Balancer Controller (LBC) a tratar o tráfego HTTP de entrada como “voltado para a internet” e encaminhá-lo para o serviço apropriado no namespace “game-2048” usando o tipo de destino “ip”. Para obter mais anotações, consulte Annotations na documentação do AWS LBC.

1. Crie um namespace do Kubernetes denominado `game-2048` com o sinalizador `--save-config`.

   ```
   kubectl create namespace game-2048 --save-config
   ```

   Você verá a seguinte saída de resposta:

   ```
   namespace/game-2048 created
   ```

2. Implante a aplicação de amostra do jogo 2048.

   ```
   kubectl apply -n game-2048 -f https://raw.githubusercontent.com/kubernetes-sigs/
   aws-load-balancer-controller/v2.8.0/docs/examples/2048/2048_full.yaml
   ```

   Esse manifesto configura uma implantação, um serviço e uma entrada do Kubernetes para o namespace `game-2048`, criando os recursos necessários para implantar e expor a aplicação `game-2048` no cluster. Isso inclui a criação de um serviço denominado `service-2048` que expõe a implantação na porta `80` e um recurso de entrada denominado `ingress-2048` que define regras de roteamento para tráfego HTTP de entrada e anotações para um Application Load Balancer (ALB) voltado para a internet. Você verá a seguinte saída de resposta:

   ```
   namespace/game-2048 configured
   deployment.apps/deployment-2048 created
   service/service-2048 created
   ingress.networking.k8s.io/ingress-2048 created
   ```

3. Execute o comando a seguir para obter o recurso de entrada para o namespace `game-2048`.

   ```
   kubectl get ingress -n game-2048
   ```

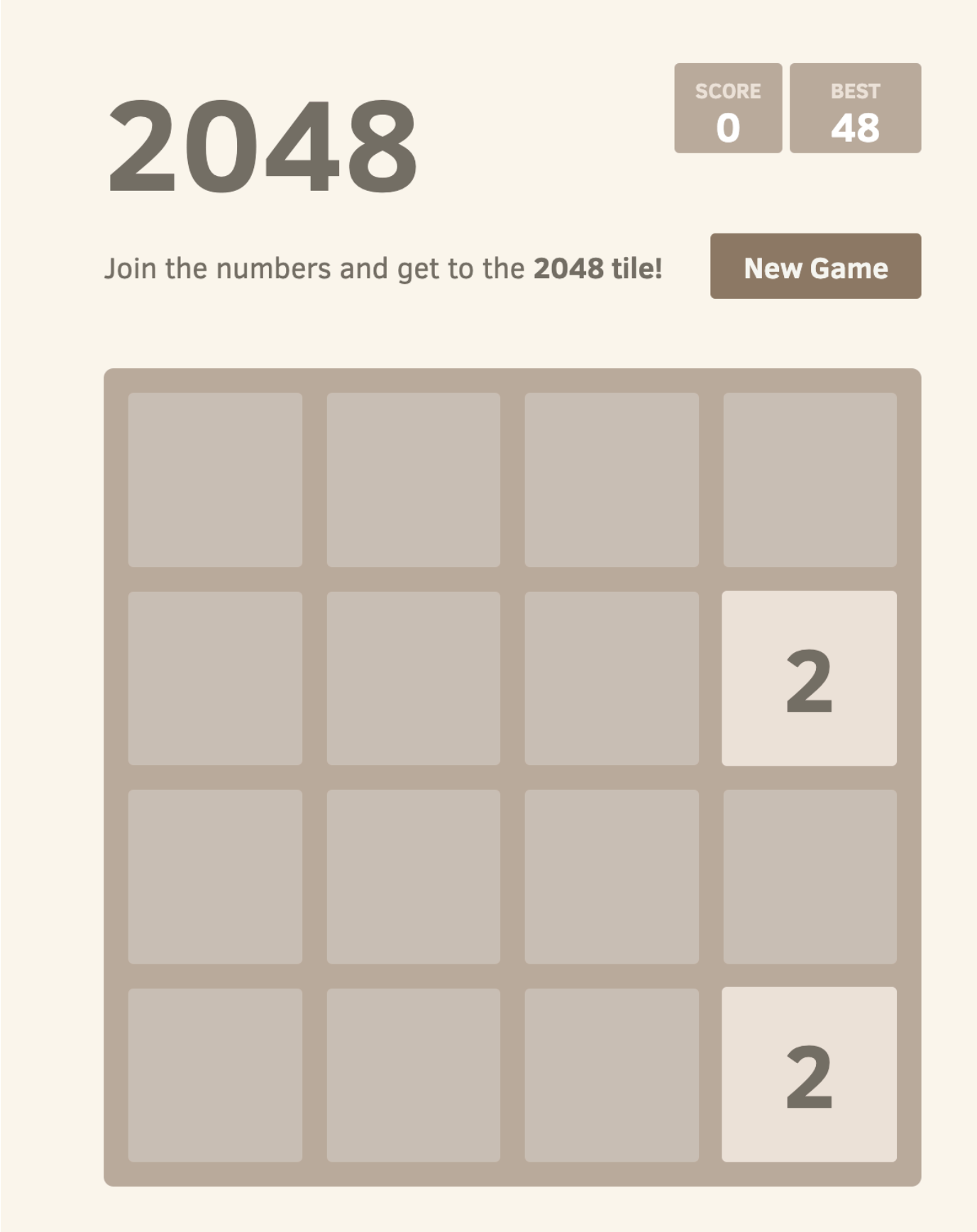
   Você verá a seguinte saída de resposta:

   ```
   NAME           CLASS    HOSTS   ADDRESS
                           PORTS   AGE
   ```

```
ingress-2048   alb     *       k8s-game2048-ingress2-eb379a0f83-378466616.region-
code.elb.amazonaws.com   80      31s
```

Você precisará esperar alguns minutos para que o Application Load Balancer (ALB) seja provisionado antes de começar as etapas a seguir.

4. Abra o navegador da Web e digite o `ADDRESS` da etapa anterior para acessar a aplicação Web. Por exemplo, `k8s-game2048-ingress2-eb379a0f83-378466616.`*`region-code`*`.elb.amazonaws.com`. Você deve ver o jogo 2048 no seu navegador. Reproduza.


2048
SCORE
0
BEST
48
Join the numbers and get to the 2048 tile!
New Game
2
2

Etapa 5: persistir os dados usando os nós do driver CSI do Amazon EBS

Agora que o jogo 2048 está instalado e em execução no seu cluster do Amazon EKS, é hora de garantir que os dados do jogo sejam persistidos com segurança usando o complemento gerenciado do driver CSI do Amazon EBS. Esse complemento foi instalado no cluster durante o processo de criação. Essa integração é essencial para preservar o progresso e os dados do jogo, mesmo quando os pods ou nós do Kubernetes são reiniciados ou substituídos.

1. Crie uma classe de armazenamento para o driver CSI do EBS:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-ebs-csi-driver/master/examples/kubernetes/dynamic-provisioning/manifests/storageclass.yaml
```

2. Crie uma solicitação de volume persistente (PVC) para solicitar armazenamento para os dados do seu jogo. Crie um arquivo denominado `ebs-pvc.yaml` e adicione o seguinte conteúdo a ele:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: game-data-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: ebs-sc
```

3. Aplique a PVC ao cluster:

```
kubectl apply -f ebs-pvc.yaml
```

   Você verá a seguinte saída de resposta:

```
persistentvolumeclaim/game-data-pvc created
```

4. Você agora precisa atualizar a implantação do jogo 2048 para usar essa PVC para armazenar dados. A implantação a seguir está configurada para usar a PVC para armazenar dados do jogo. Crie um arquivo denominado `ebs-deployment.yaml` e adicione o seguinte conteúdo a ele:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  replicas: 3  # Adjust the number of replicas as needed
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - name: app-2048
          image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 80
          volumeMounts:
            - name: game-data
              mountPath: /var/lib/2048
      volumes:
        - name: game-data
          persistentVolumeClaim:
            claimName: game-data-pvc
```

5. Aplique a implantação atualizada:

```
kubectl apply -f ebs-deployment.yaml
```

Você verá a seguinte saída de resposta:

```
deployment.apps/deployment-2048 configured
```

Com essas etapas, seu jogo 2048 no Amazon EKS agora está configurado para persistir os dados usando o driver CSI do Amazon EBS. Isso garante que o progresso e os dados do jogo estejam seguros mesmo no caso de falhas nos pods ou nós. Caso tenha gostado deste tutorial, deixe seu

feedback para que possamos fornecer mais tutoriais de início rápido específicos para casos de uso, como este.

Etapa 6: excluir clusters e nós

Após concluir o cluster e os nós criados para este tutorial, faça uma limpeza excluindo o cluster e os nós usando o comando a seguir. Caso queira realizar outras ações com esse cluster antes de limpá-lo, consulte Próximas etapas.

```
eksctl delete cluster -f ./cluster-config.yaml
```

Após a conclusão, você verá a seguinte saída de resposta:

```
2024-07-05 17:26:44 [#] all cluster resources were deleted
```

Próximas etapas

Os seguintes tópicos da documentação ajudam a estender a funcionalidade do cluster:

- A entidade principal do IAM que criou o cluster é a única entidade que pode fazer chamadas ao servidor da API do Kubernetes usando `kubectl` ou o AWS Management Console. Se quiser que outras entidades principais do IAM tenham acesso ao cluster, será necessário adicioná-los. Para ter mais informações, consulte Conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes e Permissões obrigatórias.
- Antes de implantar um cluster para uso em produção, recomendamos conhecer todas as configurações de clusters e nós. Algumas configurações (como habilitar o acesso SSH aos nós do Amazon EC2) devem ser feitas no momento de criação do cluster.
- Para aumentar a segurança do cluster, configure o plugin Amazon VPC Container Networking Interface para usar funções do IAM para contas de serviço.

Para explorar formas de criar diferentes tipos de clusters:

- Configuração do cluster do EKS na comunidade AWS

Começar a usar o Amazon EKS

Antes de ler os guias de introdução, verifique se você está configurado para usar o Amazon EKS. Para ter mais informações, consulte Configurar para usar o Amazon EKS.

Há dois manuais de introdução disponíveis para a criação de um novo cluster do Kubernetes com nós no Amazon EKS:

- Conceitos básicos do Amazon EKS: `eksctl`: esse guia de conceitos básicos ajuda você a instalar todos os recursos necessários para começar a usar o Amazon EKS com o `eksctl`, um utilitário de linha de comando simples para criar e gerenciar clusters de Kubernetes no Amazon EKS. No final do tutorial, você terá um cluster do Amazon EKS em execução no qual poderá implantar as aplicações. Essa é a forma mais rápida e simples para começar a usar o Amazon EKS.
- Conceitos básicos do Amazon EKS: AWS Management Console e AWS CLI – Este guia de conceitos básicos ajuda a criar todos os recursos necessários para começar a usar o Amazon EKS no AWS Management Console e na AWS CLI. No final do tutorial, você terá um cluster do Amazon EKS em execução no qual poderá implantar as aplicações. Neste manual, crie manualmente cada recurso necessário para um cluster do Amazon EKS. Os procedimentos fornecem visibilidade sobre como cada recurso é criado e como eles interagem uns com os outros.

Também oferecemos as seguintes referências:

- Para ver uma coleção selecionada de tutoriais práticos, consulte Navegar no Amazon EKS, na Comunidade da AWS.
- Para exemplos de código, consulte Exemplos de código para o Amazon EKS usando AWS SDKs.

Conceitos básicos do Amazon EKS: **`eksctl`**

Este guia ajuda você a criar todos os recursos necessários para começar a usar o Amazon Elastic Kubernetes Service (Amazon EKS) com o `eksctl`, um utilitário de linha de comando simples para criar e gerenciar clusters do Kubernetes no Amazon EKS. No final deste tutorial, você terá um cluster do Amazon EKS em execução, no qual poderá implantar as aplicações.

Os procedimentos neste guia criam automaticamente vários recursos que você precisa criar manualmente ao criar o cluster usando o comando AWS Management Console. Se preferir criar manualmente a maioria dos recursos e entender melhor como eles interagem entre si, use o AWS

Management Console para criar os clusters e a computação. Para obter mais informações, consulte Conceitos básicos do Amazon EKS: AWS Management Console e AWS CLI.

Pré-requisitos

Antes de iniciar este tutorial, você deve instalar e configurar as ferramentas do kubectl, eksctl e AWS CLI, conforme descrito em Configurar para usar o Amazon EKS.

Etapa 1: Criar o cluster e nós do Amazon EKS

> **Important**
>
> Para começar da forma mais simples e rápida possível, este tópico inclui etapas para criar um cluster e nós com configurações padrão. Antes de criar um cluster e nós para uso em produção, recomendamos que você se familiarize com todas as configurações e implante um cluster e nós com as configurações que atendam aos seus requisitos. Para ter mais informações, consulte Criar um cluster do Amazon EKS. e Gerenciar recursos computacionais usando nós. Algumas configurações poderão ser habilitadas apenas durante a criação do cluster e dos nós.

Você pode criar um cluster com um dos seguintes tipos de nós. Para saber mais sobre cada tipo, consulte Gerenciar recursos computacionais usando nós. Depois que o cluster for implantado, você pode adicionar outros tipos de nós.

- Fargate: Linux: selecione este tipo de nó se quiser executar aplicações do Linux no AWS Fargate. O Fargate é um mecanismo de computação com tecnologia sem servidor que permite a implantação de Pods do Kubernetes sem gerenciar instâncias do Amazon EC2.
- Nós gerenciados: Linux: selecione este tipo de nó se quiser executar aplicações do Amazon Linux em instâncias do Amazon EC2. Embora não abordado neste guia, você também pode adicionar nós autogerenciados do Windows e do Bottlerocket ao cluster.

Crie o cluster do Amazon EKS com o comando a seguir. Você pode substituir *`my-cluster`* por seus próprios valores. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster. Substitua *`region-code`* por qualquer Região da AWS compatível com

o Amazon EKS. Para obter uma lista das Regiões da AWS, consulte Endpoints e cotas do Amazon EKS no guia Referência geral da AWS.

Fargate – Linux

```
eksctl create cluster --name my-cluster --region region-code --fargate
```

Managed nodes – Linux

```
eksctl create cluster --name my-cluster --region region-code
```

A criação do cluster leva muitos minutos. Durante a criação, você verá várias linhas de saída. A última linha do resultado é semelhante ao exemplo de linha a seguir.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

`eksctl` criou um arquivo `kubectl config` em `~/.kube` ou adicionou a configuração do novo cluster dentro de um arquivo `config` em `~/.kube` no seu computador.

Após a criação do cluster ser concluída, exiba a pilha do AWS CloudFormation chamada `eksctl-my-cluster-cluster` no console do AWS CloudFormation, em https://console.aws.amazon.com/cloudformation para ver todos os recursos que você criou.

Etapa 2: Visualizar recursos do Kubernetes

1. Exiba os nós de cluster.

   ```
   kubectl get nodes -o wide
   ```

   Veja um exemplo de saída abaixo.

   Fargate – Linux

   ```
   NAME                                                STATUS   ROLES    AGE
      VERSION              INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-
   VERSION                   CONTAINER-RUNTIME
   ```

```
fargate-ip-192-0-2-0.region-code.compute.internal   Ready    <none>
  8m3s    v1.2.3-eks-1234567   192.0.2.0     <none>        Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
fargate-ip-192-0-2-1.region-code.compute.internal   Ready    <none>
  7m30s   v1.2.3-eks-1234567   192-0-2-1     <none>        Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Managed nodes – Linux

```
NAME                                          STATUS   ROLES    AGE    VERSION
          INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
       CONTAINER-RUNTIME
ip-192-0-2-0.region-code.compute.internal   Ready    <none>   6m7s
   v1.2.3-eks-1234567   192.0.2.0     192.0.2.2     Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
ip-192-0-2-1.region-code.compute.internal   Ready    <none>   6m4s
   v1.2.3-eks-1234567   192.0.2.1     192.0.2.3     Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Para obter mais informações sobre o que é visualizado na saída, consulte Visualizar os recursos do Kubernetes.

2. Exiba as workloads em execução no cluster.

```
kubectl get pods -A -o wide
```

Veja um exemplo de saída abaixo.

Fargate – Linux

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE   IP
          NODE                                                NOMINATED NODE
 READINESS GATES
kube-system   coredns-1234567890-abcde   1/1     Running   0          18m
  192.0.2.0   fargate-ip-192-0-2-0.region-code.compute.internal   <none>
   <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          18m
  192.0.2.1   fargate-ip-192-0-2-1.region-code.compute.internal   <none>
   <none>
```

Managed nodes – Linux

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
          NODE                                            NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-12345                1/1     Running   0          7m43s
   192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
kube-system   aws-node-67890                1/1     Running   0          7m46s
   192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-abcde      1/1     Running   0          14m
   192.0.2.3   ip-192-0-2-3.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-12345      1/1     Running   0          14m
   192.0.2.4   ip-192-0-2-4.region-code.compute.internal   <none>
 <none>
kube-system   kube-proxy-12345              1/1     Running   0          7m46s
   192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   kube-proxy-67890              1/1     Running   0          7m43s
   192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
```

Para obter mais informações sobre o que é visualizado na saída, consulte Visualizar os recursos do Kubernetes.

Etapa 3: excluir clusters e nós

Após concluir o cluster e os nós criados para este tutorial, faça uma limpeza excluindo o cluster e os nós usando o comando a seguir. Se você quiser realizar outras ações com esse cluster antes de limpá-lo, consulte Próximas etapas.

```
eksctl delete cluster --name my-cluster --region region-code
```

Próximas etapas

Os seguintes tópicos de documentação ajudam a estender a funcionalidade do seu cluster:

- Implante uma aplicação de exemplo no cluster.
- A entidade principal do IAM que criou o cluster é a única entidade que pode fazer chamadas ao servidor da API do Kubernetes usando `kubectl` ou o AWS Management Console. Se quiser que outras entidades principais do IAM tenham acesso ao cluster, será necessário adicioná-los. Para ter mais informações, consulte Conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes e Permissões obrigatórias.
- Antes de implantar um cluster para uso em produção, recomendamos conhecer todas as configurações de clusters e nós. Algumas configurações (como habilitar o acesso SSH aos nós do Amazon EC2) devem ser feitas no momento de criação do cluster.
- Para aumentar a segurança do cluster, configure o plugin Amazon VPC Container Networking Interface para usar funções do IAM para contas de serviço.

Conceitos básicos do Amazon EKS: AWS Management Console e AWS CLI

Este manual ajuda a criar todos os recursos necessários para começar a usar o Amazon Elastic Kubernetes Service (Amazon EKS) com o AWS Management Console e o AWS CLI. Neste manual, você criará manualmente cada recurso. No final deste tutorial, você terá um cluster do Amazon EKS em execução, no qual poderá implantar as aplicações.

Os procedimentos deste guia fornecem visibilidade completa sobre como cada recurso é criado e como os recursos interagem entre si. Se você preferir que a maioria dos recursos seja criada automaticamente, use a CLI `eksctl` para criar os nós e o cluster. Para obter mais informações, consulte Conceitos básicos do Amazon EKS: `eksctl`.

Pré-requisitos

Antes de iniciar este tutorial, você deve instalar e configurar as ferramentas a seguir e os recursos necessários para criar e gerenciar um cluster do Amazon EKS.

- AWS CLI – Uma ferramenta de linha de comando para trabalhar com os serviços AWS, incluindo o Amazon EKS. Para obter mais informações, consulte Installing, updating, and uninstalling the AWS CLI (Instalar, atualizar e desinstalar a) no Manual do usuário da AWS Command Line Interface. Depois de instalar a AWS CLI, recomendamos que você também a configure. Para obter mais informações, consulte Quick configuration with `aws configure` (Configuração rápida com) no Manual do usuário do AWS Command Line Interface.

- **`kubectl`**: uma ferramenta de linha de comando para trabalhar com clusters do Kubernetes. Para obter mais informações, consulte Configurar o `kubectl` e o `eksctl`.
- Permissões obrigatórias do IAM: a entidade principal de segurança do IAM que você está usando deve ter permissões para trabalhar com perfis do IAM para o Amazon EKS e funções vinculadas ao serviço, AWS CloudFormation, uma VPC e recursos relacionados. Para obter mais informações, consulte Ações, recursos e chaves de condição do Amazon Elastic Kubernetes Service e Usar funções vinculadas a serviço no Guia do usuário do IAM. Você deve concluir todas as etapas deste manual como o mesmo usuário. Execute o seguinte comando para verificar o usuário atual:

```
aws sts get-caller-identity
```

- Convém concluir as etapas neste tópico em um shell Bash. Se não estiver utilizando um shell Bash, alguns comandos de script, como caracteres de continuação de linha e a forma como as variáveis são definidas e utilizadas, exigirão o ajuste do seu shell. Além disso, as regras de citação e de escape do seu shell podem ser diferentes. Para obter mais informações, consulte Uso de aspas com strings na AWS CLI no Guia do usuário da AWS Command Line Interface.

Etapa 1: Criar o cluster do Amazon EKS

> **Important**
>
> Para começar da forma mais simples e rápida possível, este tópico inclui etapas para criar um cluster com configurações padrão. Antes de criar um cluster para uso em produção, recomendamos que você se familiarize com todas as configurações e implante um cluster com as configurações que atendam aos seus requisitos. Para obter mais informações, consulte Criar um cluster do Amazon EKS.. Algumas configurações poderão ser habilitadas apenas durante a criação do cluster.

Para criar um cluster

1. Crie uma Amazon VPC com sub-redes públicas e privadas que atenda aos requisitos do Amazon EKS. Substitua *`region-code`* por qualquer Região da AWS compatível com o Amazon EKS. Para obter uma lista das Regiões da AWS, consulte Endpoints e cotas do Amazon EKS no Guia de referência geral da AWS. Você pode substituir *`my-eks-vpc-stack`* por qualquer nome que escolher.

```
aws cloudformation create-stack \
  --region region-code \
  --stack-name my-eks-vpc-stack \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

> **Tip**
>
> Para obter uma lista de todos os recursos que o comando anterior cria, abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation. Escolha a pilha *my-eks-vpc-stack* e depois a guia Resources (Recursos).

2. Crie um perfil do IAM de cluster e anexe a ele a política gerenciada pelo IAM no Amazon EKS requerida. Os clusters do Kubernetes gerenciados pelo Amazon EKS fazem chamadas para outros serviços da AWS em seu nome para gerenciar os recursos que você usa com o serviço.

   a. Copie o conteúdo a seguir em um arquivo denominado *eks-cluster-role-trust-policy.json*.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "eks.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   b. Crie a função.

   ```
   aws iam create-role \
     --role-name myAmazonEKSClusterRole \
     --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
   ```

   c. Anexe a política de IAM gerenciada pelo Amazon EKS à função.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name myAmazonEKSClusterRole
```

3. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

   Verifique se a Região da AWS exibida no canto superior direito do seu console é a Região da AWS em que você deseja criar o cluster. Se não for, escolha a lista suspensa ao lado do nome da Região da AWS e, em seguida, a Região da AWS que deseja usar.

4. Escolha Add cluster (Adicionar cluster) e, em seguida, Create (Criar). Caso não visualize essa opção, escolha Clusters no painel de navegação à esquerda.

5. Na página Configure cluster (Configurar cluster), faça o seguinte:

   a. Insira um nome para o seu cluster, como **`my-cluster`**. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.

   b. Em Cluster Service Role (Função de serviço do cluster), escolha *`myAmazonEKSClusterRole`*.

   c. Deixe as configurações restantes com seus valores padrão e escolha Next (Próximo).

6. Na página Specify networking (Especificar redes), faça o seguinte:

   a. Escolha o ID da VPC que você criou em uma etapa anterior na lista suspensa VPC. É algo como *`vpc-00x0000x000x0x000`* | *`my-eks-vpc-stack-VPC`*.

   b. Deixe as configurações restantes com seus valores padrão e escolha Next (Próximo).

7. Na página Configurar observabilidade, escolha Próximo.

8. Na página Selecionar complementos, selecione Avançar.

   Para obter mais informações sobre complementos, consulte Complementos do Amazon EKS.

9. Na página Definir as configurações dos complementos selecionados, escolha Avançar.

10. Na página Review and create (Revisar e criar), escolha Create (Criar).

    À direita do nome do cluster, o status do cluster será Creating (Em criação) por vários minutos, até que o processo de provisionamento do cluster seja concluído. Não passe para a próxima etapa até que o status seja Active (Ativo).

> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte Insufficient capacity (Capacidade insuficiente).

Etapa 2: Configurar o computador para se comunicar com seu cluster

Nesta seção, você cria um arquivo `kubeconfig` para o cluster. As configurações neste arquivo ativam a opção CLI do `kubectl` para se comunicar com o cluster.

Para configurar o computador para se comunicar com o cluster

1. Crie um arquivo `kubeconfig` para o cluster. Substitua *`region-code`* pela Região da AWS na qual você criou o cluster. Substitua o *`my-cluster`* pelo nome do cluster.

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   Por padrão, um arquivo `~/.kube` é criado em `config` ou o novo cluster é adicionado a um arquivo `config` existente no `~/.kube`.
2. Teste a configuração.

   ```
   kubectl get svc
   ```

   > **Note**
   >
   > Se você receber qualquer erro de autorização ou de tipo de recurso, consulte Acesso negado ou não autorizado (`kubectl`) no tópico de solução de problemas.

   Veja um exemplo de saída abaixo.

   ```
   NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   ```

```
svc/kubernetes    ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

Etapa 3: criar nós

> **Important**
>
> Para começar da forma mais simples e rápida possível, este tópico inclui etapas para criar nós com configurações padrão. Antes de criar nós para uso em produção, recomendamos que você se familiarize com todas as configurações e implante nós com as configurações que atendam aos seus requisitos. Para obter mais informações, consulte Gerenciar recursos computacionais usando nós. Algumas configurações poderão ser habilitadas apenas durante a criação dos nós.

Você pode criar um cluster com um dos seguintes tipos de nós. Para saber mais sobre cada tipo, consulte Gerenciar recursos computacionais usando nós. Depois que o cluster for implantado, você pode adicionar outros tipos de nós.

- Fargate: Linux: escolha este tipo de nó se quiser executar aplicações do Linux no AWS Fargate. O Fargate é um mecanismo de computação com tecnologia sem servidor que permite a implantação de Pods do Kubernetes sem gerenciar instâncias do Amazon EC2.
- Nós gerenciados: Linux: escolha este tipo de nó se quiser executar aplicações do Amazon Linux em instâncias do Amazon EC2. Embora não abordado neste guia, você também pode adicionar nós autogerenciados do Windows e Bottlerocket ao cluster.

Fargate – Linux

Criar um perfil do Fargate. Quando os Pods do Kubernetes são implantados com critérios que correspondem aos critérios definidos no perfil, Pods são implantados no Fargate.

Para criar um perfil do Fargate

1. Crie uma função do IAM de cluster e associe a política gerenciada do Amazon EKS IAM a ela. Quando o cluster cria Pods na infraestrutura do Fargate, os componentes que estão em execução nessa infraestrutura do Fargate devem fazer as chamadas para as APIs da AWS em seu nome. Isso ocorre para que eles possam realizar ações como extrair imagens de

contêiner do Amazon ECR ou encaminhar logs para outros serviços da AWS. A função de execução de Pod do Amazon EKS fornece as permissões do IAM para isso.

a. Copie o conteúdo a seguir em um arquivo denominado *pod-execution-role-trust-policy*.json. Substitua *region-code* pela Região da AWS em que seu cluster está localizado. Se quiser utilizar a mesma função em todas as Regiões da AWS da sua conta, substitua *region-code* por *. Substitua *111122223333* pelo ID da conta e *my-cluster* pelo nome do cluster. Se quiser usar a mesma função para todos os clusters da sua conta, substitua *my-cluster* por *.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Crie um perfil do IAM de execução de Pod.

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-
policy.json"
```

c. Anexe a política de IAM gerenciada pelo Amazon EKS à função.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/
AmazonEKSFargatePodExecutionRolePolicy \
```

```
    --role-name AmazonEKSFargatePodExecutionRole
```

2. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
3. Na página Clusters, escolha o cluster *my-cluster*.
4. Na página ***my-cluster***, faça o seguinte:
   a. Escolha a guia Compute (Computação).
   b. Em Fargate Profiles (Perfis do Fargate), escolha Add Fargate profile (Adicionar perfil do Fargate).
5. Na página Configure Fargate Profile (Configurar o perfil do Fargate), faça o seguinte:
   a. Em Name (Nome), insira um nome exclusivo para o perfil do Fargate, como ***my-profile***.
   b. Em Pod execution role (Função de execução de pod), escolha a função AmazonEKSFargatePodExecutionRole criada em uma etapa anterior.
   c. Escolha a lista suspensa Subnets (Sub-redes) e desmarque qualquer sub-rede que tenha `Public` no nome. Somente sub-redes privadas são compatíveis com Pods em execução no Fargate.
   d. Escolha Próximo.
6. Na página Configure Pod selection (Configurar a seleção de), faça o seguinte:
   a. Em Namespace, insira **`default`**.
   b. Escolha Próximo.
7. Na página Review and create (Revisar e criar), revise as informações do perfil do Fargate e selecione Create (Criar).
8. Após alguns minutos, o Status na seção Fargate Profile configuration (Configuração do perfil do Fargate) será alterada de Creating (Em criação) para Active (Ativo). Não passe para a próxima etapa até que o status seja Active (Ativo).
9. Se você planeja implantar todos os Pods no Fargate (nenhum em nós do Amazon EC2), use o procedimento a seguir para criar outro perfil do Fargate e executar o resolvedor de nomes padrão (CoreDNS) no Fargate.

> **Note**
>
> Se você não fizer isso, não obterá nenhum nó no momento.

a. Na página Fargate Profile (Perfil do Fargate), escolha *my-profile*.

b. Em Fargate profiles (Perfis do Fargate), escolha Add Fargate profile (Adicionar perfil do Fargate).

c. Em Nome, digite ***CoreDNS***.

d. Em Pod execution role (Função de execução de pod), escolha a função AmazonEKSFargatePodExecutionRole criada em uma etapa anterior.

e. Escolha a lista suspensa Subnets (Sub-redes) e desmarque qualquer sub-rede que tenha `Public` no nome. Somente sub-redes privadas são compatíveis com os Pods em execução no Fargate.

f. Escolha Próximo.

g. Em Namespace, insira **`kube-system`**.

h. Escolha Match labels (Rótulos de correspondência) e, em seguida, Add label (Adicionar rótulo).

i. Insira **`k8s-app`** em Key (Chave) e **`kube-dns`** em value (valor). Isso é necessário para implantar o resolvedor de nomes padrão (CoreDNS) no Fargate.

j. Escolha Próximo.

k. Na página Review and create (Revisar e criar), revise as informações do perfil do Fargate e selecione Create (Criar).

l. Depois, use o comando a seguir para remover a anotação padrão `eks.amazonaws.com/compute-type : ec2` do CoreDNS e do Pods.

```
kubectl patch deployment coredns \
    -n kube-system \
    --type json \
    -p='[{"op": "remove", "path": "/spec/template/metadata/annotations/
eks.amazonaws.com~1compute-type"}]'
```

> **Note**
>
> O sistema cria e implanta dois nós com base no rótulo de perfil do Fargate adicionado por você. Você não verá nada listado em Node Groups (Grupos de nós)

porque eles não são aplicáveis aos nós do Fargate, mas encontrará os novos nós listados na guia Overview (Visão geral).

Managed nodes – Linux

Crie um grupo de nós gerenciados, especificando as sub-redes e a função do IAM do nó que você criou nas etapas anteriores.

Para criar o grupo de nós gerenciados do Amazon EC2 Linux

1. Crie uma função do IAM do nó e associe a política gerenciada do Amazon EKS IAM a ela. O daemon `kubelet` do nó do Amazon EKS chama as APIs da AWS em seu nome. Os nós recebem permissões para essas chamadas de API por meio de um perfil de instância do IAM e políticas associadas.

   a. Copie o conteúdo a seguir em um arquivo denominado *node-role-trust-policy.json*.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "ec2.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   b. Crie a função do IAM para o nó.

   ```
   aws iam create-role \
     --role-name myAmazonEKSNodeRole \
     --assume-role-policy-document file://"node-role-trust-policy.json"
   ```

   c. Anexe à função as políticas do IAM gerenciadas necessárias.

   ```
   aws iam attach-role-policy \
   ```

```
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name myAmazonEKSNodeRole
```

2. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
3. Escolha o nome do cluster criado por você no Etapa 1: Criar o cluster do Amazon EKS, por exemplo, *my-cluster*.
4. Na página ***my-cluster***, faça o seguinte:
   a. Escolha a guia Compute (Computação).
   b. Escolha Add Node Group (Adicionar grupo de nós).
5. Na página Configure Node Group (Configurar grupo de nós), faça o seguinte:
   a. Em Name (Nome), insira um nome exclusivo para o seu grupo de nós gerenciados, como ***my-nodegroup***. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.
   b. Em Node IAM role name (Nome da função do IAM de nós), escolha a função *myAmazonEKSNodeRole* que você criou em uma etapa anterior. Recomendamos que cada grupo de nós use sua própria função exclusiva do IAM.
   c. Escolha Próximo.
6. Na página Set compute and scaling configuration (Definir configurações de computação e escalabilidade), aceite os valores padrão e escolha Next (Próximo).
7. Na página Specify networking (Especificar redes), aceite os valores padrão e escolha Next (Próximo).
8. Na página Review and create (Revisar e criar), reveja a configuração do grupo de nós gerenciados e escolha Create (Criar).
9. Após alguns minutos, o Status em Node Group configuration(Configuração do grupo de nós) mudará de Creating (Em criação) para Active (Ativo). Não passe para a próxima etapa até que o status seja Active (Ativo).

Etapa 4: visualizar recursos

Você pode exibir os nós e as workloads do Kubernetes.

Para visualizar nós e workloads

1. No painel de navegação à esquerda, escolha Clusters. Escolha o nome do cluster criado por você, por exemplo, *my-cluster*, na lista de Clusters.
2. Na página ***my-cluster***, escolha o seguinte:
    a. Guia Compute (Computação): você verá a lista Nodes (Nós) de nós que foram implantados para o cluster. Você pode escolher o nome de um nó para consultar mais informações sobre ele.
    b. Guia Resources (Recursos): você verá todos os recursos Kubernetes implantados por padrão em um cluster do Amazon EKS. Selecione qualquer tipo de recurso no console para obter mais informações sobre ele.

Etapa 5: excluir recursos

Após concluir o cluster e os nós criados para este tutorial, exclua os recursos que você criou. Se quiser realizar outras ações com esse cluster antes de excluir os recursos, consulte Próximas etapas.

Para excluir os recursos que você criou neste guia

1. Exclua todos os grupos de nós ou perfis do Fargate que criou.
    a. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
    b. No painel de navegação à esquerda, escolha Clusters. Na lista de clusters, escolha *my-cluster*.
    c. Escolha a guia Compute (Computação).
    d. Se você criou um grupo de nós, scolha o grupo de nós *my-nodegroup* e depois escolha Delete (Excluir). Insira ***my-nodegroup*** e escolha Delete (Excluir).
    e. Escolha-o para cada perfil do Fargate criado por você e, então, escolha Delete (Excluir). Insira o nome do perfil e, então, escolha Delete (Excluir).

> **Note**
>
> Ao excluir um segundo perfil do Fargate, talvez seja necessário esperar a conclusão de exclusão do primeiro.

f. Não continue até que o grupo de nós ou perfis do Fargate estejam excluído(s).

2. Excluir o cluster.

   a. No painel de navegação à esquerda, escolha Clusters. Na lista de clusters, escolha *my-cluster*.

   b. Escolha Delete Cluster (Excluir cluster).

   c. Insira ***my-cluster*** e escolha Delete (Excluir). Não continue até que o cluster seja excluído.

3. Exclua a pilha do AWS CloudFormation da VPC criada.

   a. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.

   b. Escolha a pilha *my-eks-vpc-stack* e depois Delete (Excluir).

   c. Na caixa de diálogo de confirmação Delete ***my-eks-vpc-stack*** (Excluir o my-eks-vpc-stack), escolha Delete stack (Excluir pilha).

4. Exclua as funções do IAM criadas.

   a. Abra o console do IAM em https://console.aws.amazon.com/iam/.

   b. No painel de navegação à esquerda, escolha Roles.

   c. Na lista, selecione cada função criada (***myAmazonEKSClusterRole***, bem como AmazonEKSFargatePodExecutionRole ou *myAmazonEKSNodeRole*). Escolha Delete (Excluir), insira o texto de confirmação solicitado e escolha Delete (Excluir).

Próximas etapas

Os seguintes tópicos de documentação ajudam a estender a funcionalidade do seu cluster:

- A entidade principal do IAM que criou o cluster é a única entidade que pode fazer chamadas ao servidor da API do Kubernetes usando `kubectl` ou o AWS Management Console. Se quiser que outras entidades principais do IAM tenham acesso ao cluster, será necessário adicioná-los.

Para ter mais informações, consulte Conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes e Permissões obrigatórias.

- Implante uma aplicação de exemplo no cluster.
- Antes de implantar um cluster para uso em produção, recomendamos conhecer todas as configurações de clusters e nós. Algumas configurações (como habilitar o acesso SSH aos nós do Amazon EC2) devem ser feitas no momento de criação do cluster.
- Para aumentar a segurança do cluster, configure o plugin Amazon VPC Container Networking Interface para usar funções do IAM para contas de serviço.

Conhecer o Amazon EKS por meio de exemplos

Visão geral

Este Guia do usuário do Amazon EKS contém procedimentos de uso geral para criar seu primeiro cluster do EKS na linha de comando ou no Console de Gerenciamento da AWS e é uma referência confiável para todos os principais componentes do Amazon EKS. No entanto, como administrador ou desenvolvedor de clusters do Amazon EKS, você pode adquirir uma compreensão mais profunda do Amazon EKS percorrendo os caminhos de aprendizado que existem em sites que não constam neste guia. Esses sites podem ajudar a:

- Configurar tipos específicos de clusters. Tipos específicos de clusters podem ser baseados em seus tipos de workloads ou requisitos de segurança. Por exemplo, você talvez queira ajustar um cluster para executar workloads em lote, de machine learning ou de computação intensiva.
- Melhorar seus clusters. Você pode adicionar recursos avançados ao seu cluster para fornecer elementos como observabilidade, armazenamento flexível, escalonamento automático ou rede de cluster especializada.
- Automatizar atualizações. Usando recursos como o GitOps, você pode configurar o provisionamento automático da infraestrutura de cluster e das workloads com base nas alterações que ocorrem nesses componentes nos seus repositórios Git.
- Usar ferramentas avançadas de configuração de clusters. Embora o `eksctl` ofereça uma maneira rápida de criar um cluster, existem outras ferramentas que podem facilitar a configuração e a atualização de clusters mais complexos. Isso inclui ferramentas como o Terraform e CloudFormation.

Para começar sua trajetória de aprendizado do Amazon EKS, recomendo que visite alguns dos sites descritos nesta página. Se você tiver problemas ao longo do caminho, também há recursos para ajudar a superá-los. Por exemplo, o Centro de Conhecimento re:Post permite pesquisar problemas de suporte relacionados ao Amazon EKS no banco de dados de suporte. Além disso, o Guia de práticas recomendadas do Amazon EKS oferece dicas sobre as melhores maneiras de configurar seus clusters do nível de produção.

Workshop do Amazon EKS

Começando com uma compreensão básica do Kubernetes e de contêineres, o Workshop do Amazon EKS é uma plataforma de aprendizado para orientar um administrador de clusters com relação a recursos importantes do Amazon EKS. Aqui estão algumas maneiras pelas quais você pode participar do workshop do Amazon EKS:

- Princípios básicos do Amazon EKS: assista ao vídeo na página de introdução para saber como o Amazon EKS implementa os recursos do Kubernetes na Nuvem AWS. Se você precisar de uma compreensão ainda mais básica do Kubernetes, assista ao vídeo What is Kubernetes.
- Configuração do Amazon EKS: se você tiver uma conta da AWS, a seção Configuração ajudará você a configurar um ambiente do CloudShell para criar um cluster. Ele oferece a opção de eksctl (uma linha de comando simples para criação de clusters) e o Terraform (uma abordagem mais baseada em infraestrutura como código para criar um cluster) para criar seu cluster do Amazon EKS.
- Conceitos básicos do Amazon EKS: experimente uma loja Web simples na seção Sample application. Você pode usá-la em todos os outros exercícios. Nesta seção, você também pode saber mais como empacotar imagens de contêiner e como os microsserviços são gerenciados usando pods do Kubernetess, implantações, serviços, StatefulSets e namespaces. Em seguida, use o Kustomize para implantar alterações nos manifestos do Kubernetes.
- Fundamentos do Amazon EKS: usando recursos da AWS, como o AWS Load Balancer Controller, o workshop mostra como expor suas aplicações para o mundo externo. Para armazenamento, o workshop mostra como usar o Amazon EBS para armazenamento em blocos, o Amazon EFS para armazenamento de sistemas de arquivos e o Amazon FSx para NetApp ONTAP para gerenciar sistemas de arquivos ONTAP na AWS. Para gerenciamento de nós, o workshop ajuda você a configurar Grupos de nós gerenciados.
- Recursos avançados do Amazon EKS: recursos mais avançados oferecidos no workshop do Amazon EKS incluem laboratórios para configurar:

- Escalonamento automático: inclui escalonamento automático de nós (com o Autoscaler de clusters ou o Karpenter) e escalonamento automático de workloads (com o Autoscaler horizontal de pods e o Autoscaler proporcional de clusters).
- Observabilidade: saiba mais sobre Registro em log, OpenSearch, Container Insights no Amazon EKS e Visibilidade de custos com o Kubecost em um conjunto de laboratórios de observabilidade.
- Segurança: esse conjunto de Laboratórios de segurança permite que você explore o Gerenciamento de segredos, o Amazon GuardDuty, Padrões de segurança de pods e o Gerenciamento de políticas do Kyverno.
- Rede: saiba mais sobre os recursos de rede para o Amazon EKS nos laboratórios de Rede que incluem o Amazon VPC CNI (compatível com plug-ins de rede) e o Amazon VPC Lattice (para configurar clusters em VCs e contas de usuário).
- Automação: os laboratórios de Automação orientam você pelos métodos do GitOps para gerenciar seus clusters e projetos, como Controladores da AWS para Kubernetes e Crossplane para gerenciar ambientes de gerenciamento do Amazon EKS.

Tutoriais práticos de configuração de clusters do Amazon EKS

Um conjunto de tutoriais de configuração de clusters do Amazon EKS no site da comunidade AWS pode ajudar você a criar clusters do Amazon EKS para finalidades especiais e aprimorá-los de várias maneiras. Os tutoriais são divididos em três tipos diferentes:

Criar clusters

Estes tutoriais ajudam você a criar clusters que podem ser usados para finalidades especiais. Essas finalidades especiais incluem a capacidade de executar:

- Aplicações globalmente escaláveis baseadas em IPv6
- Tarefas assíncronas em lote
- Microsserviços de alto tráfego
- Escalonamento automático com o Karpenter no Fargate
- Workloads financeiras
- Grupos de nós gerenciados pelo Windows

Aprimorar clusters

Depois de ter um cluster disponível, você pode estender e aprimorar esse cluster de forma a permitir que ele execute workloads especializadas e também aprimore os clusters. Esses tutoriais incluem maneiras de:

- Fornecer soluções de armazenamento com o EFS CSI
- Fornecer armazenamento dinâmico de banco de dados com o EBS CSI
- Expor aplicações em clusters IPv4 usando o AWS Load Balancer Controller
- Expor aplicações em clusters IPv6 usando o AWS Load Balancer Controller

Otimizar serviços da AWS

Ao usar esses tutoriais, você pode integrar melhor os clusters aos serviços da AWS. Nesses tutoriais você encontra ajuda relacionada aos seguintes tópicos:

- Gerenciamento de registros DNS para microsserviços com ExternalDNS
- Monitoramento de aplicações com o CloudWatch
- Gerenciamento de tarefas assíncronas com armazenamento do SQS e EFS
- Consumo de segredos do AWS Secrets Manager das workloads
- Configuração do mTLS com o Fargate, NGINX e ACM PCA

Amostras do Amazon EKS

O repositório de Amostras do Amazon Amazon EKS armazena manifestos para uso com o Amazon EKS. Esses manifestos oferecem a oportunidade de experimentar diferentes tipos de aplicações no Amazon EKS ou criar tipos específicos de clusters do Amazon EKS. As amostras incluem manifestos para:

- Criar um cluster do Fargate do AWS Amazon EKS
- Criar um cluster com um perfil do IAM existente
- Adicionar um grupo de nós gerenciados do Ubuntu a um cluster
- Fazer backup e restaurare o armazenamento do pod com snapshots de volume
- Recuperar volumes do EBS montados como PVCs com várias contas
- Ativar o protocolo proxy para o controlador de entrada NGINX com Classic Load Balancers
- Configurar o registro em log no Fargate para AWS OpenSearch

- Executar o SDK para Python com um provedor de identidade federado da Web
- Implantar uma amostra de aplicação em um controlador NFS CSI
- Usar snapshots de volume para StatefulSets
- Implantar pods em nós em diferentes zonas de disponibilidade

Lembre-se de que essas amostras são apenas para fins de aprendizado e teste e não devem ser usadas na produção.

Tutoriais do AWS

O site AWS Tutoriais publica alguns tutoriais do Amazon EKS, mas também oferece uma ferramenta de pesquisa para encontrar outros tutoriais publicados em sites da AWS (como o site da comunidade AWS). Os tutoriais do Amazon EKS publicados diretamente nesse site incluem:

- Deploy a Container Web App on Amazon EKS
- Run Kubernetes clusters for less (Amazon EKS and Spot instances)
- How to cost optimize Jenkins jobs on Kubernetes

Workshop para desenvolvedores

Se você é um desenvolvedor de software e deseja criar ou refatorar aplicações para execução no Amazon EKS, o workshop para desenvolvedores do Amazon EKS é um bom lugar para começar. O workshop não só ajuda você a criar aplicações em contêineres, mas também a implantar esses contêineres em um registro de contêineres (ECR) e, depois disso, em um cluster do Amazon EKS.

Comece com o Workshop Amazon EKS Python para analisar o processo de refatoração de uma aplicação Python e, em seguida, configure seu ambiente de desenvolvimento para se preparar para a implantação da aplicação. Percorra as seções sobre contêineres, Kubernetes e o Amazon EKS para se preparar para executar suas aplicações em contêineres nesses ambientes.

Workshop do Terraform

Embora o `eksctl` seja uma ferramenta simples para criar um cluster, para tipos mais complexos de implantação de infraestrutura como código do Amazon EKS, o Terraform é uma ferramenta popular de criação e gerenciamento de clusters do Amazon EKS. O Workshop Terraform Amazon EKS ensina como usar o Terraform para criar uma AWS VPC, criar clusters do Amazon EKS e adicionar

aprimoramentos opcionais ao seu cluster. Em especial, há uma seção para criar um cluster privado do Amazon EKS

Treinamento do AWS Amazon EKS

A AWS oferece treinamento formal para saber mais sobre o Amazon EKS. Um curso de treinamento de três dias, Running Containers on Amazon Elastic Kubernetes Service, ensina:

- Princípios básicos do Kubernetes e do Amazon EKS
- Como criar clusters do Amazon EKS
- Proteção do Amazon EKS com autorização RBAC do Kubernets e do AWS IAM
- Ferramentas de automação do GitOps
- Ferramentas de monitoramento
- Técnicas para melhorar o custo, a eficiência e a resiliência

Organize workloads com clusters do Amazon EKS

Um cluster do Amazon EKS consiste em dois componentes principais:

- O plano de controle do Amazon EKS
- Os nós do Amazon EKS que estão registrados no plano de controle

O ambiente de gerenciamento do Amazon EKS consiste em nós de plano de controle que executam o software Kubernetes, como `etcd` e o servidor de API do Kubernetes. O ambiente de gerenciamento é executado em uma conta gerenciada pela AWS, e a API do Kubernetes é exposta por meio do endpoint do Amazon EKS associado ao cluster. Cada ambiente de gerenciamento de cluster do Amazon EKS é um locatário único e exclusivo, e é executado em seu próprio conjunto de instâncias do Amazon EC2.

Todos os dados armazenados pelo `etcd` e pelos volumes associados do Amazon EBS são criptografados usando o AWS KMS. O plano de controle do cluster é provisionado em várias zonas de disponibilidade e liderado por um Network Load Balancer do Elastic Load Balancing. O Amazon EKS também provisiona interfaces de rede elástica nas sub-redes da VPC para fornecer conectividade das instâncias do plano de controle com os nós (por exemplo, para oferecer suporte aos fluxos de dados `kubectl exec logs proxy`).

Important

No ambiente Amazon EKS, o armazenamento `etcd` é limitado a 8 GB de acordo com a orientação de upstream. Você pode monitorar uma métrica para o tamanho atual do banco de dados executando o comando a seguir. Se seu cluster tiver uma versão `1.28` do Kubernetes abaixo, substitua *`apiserver_storage_size_bytes`* pelo seguinte:

- Versão `1.27` e `1.26` do Kubernetes: **`apiserver_storage_db_total_size_in_bytes`**
- Versão `1.25` e anterior do Kubernetes: **`etcd_db_total_size_in_bytes`**

```
kubectl get --raw=/metrics | grep "apiserver_storage_size_bytes"
```

Os nós do Amazon EKS são executados na conta da AWS e conectam-se ao plano de controle do cluster por meio do endpoint do servidor de API e de um arquivo de certificado que é criado para o cluster.

> **Note**
>
> - Você pode descobrir como os diferentes componentes do Amazon EKS funcionam no Rede Amazon EKS.
> - Para clusters conectados, consulte Conecte um cluster do Kubernetes a um console de gerenciamento do Amazon EKS com o Amazon EKS Connector.

Tópicos

Criar um cluster do Amazon EKS.

Este tópico dá uma visão geral das opções disponíveis e descreve o que deve ser considerado ao criar um cluster do Amazon EKS. Caso precise criar um cluster em um AWS Outpost, consulte Criar clusters locais do Amazon EKS no AWS Outposts para garantir alta disponibilidade. Se esta for a primeira vez que você cria um cluster do Amazon EKS, é recomendável seguir nossos guias

Começar a usar o Amazon EKS. Esses guias ajudam a criar um cluster simples e padrão sem expandir para todas as opções que estão disponíveis.

> **Note**
>
> Novo: você pode desabilitar a instalação de complementos padrão do cluster, como `vpc-cni`. Verifique as instruções da CLI.

Pré-requisitos

- Uma VPC e sub-redes existentes que atendem aos requisitos do Amazon EKS. Antes de implantar um cluster para uso em ambientes de produção, convém ter uma compreensão integral dos requisitos da VPC e da sub-rede. Se não tiver uma VPC e sub-redes, você poderá criá-las utilizando um modelo AWS CloudFormation fornecido pelo Amazon EKS.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.
- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.
- Uma entidade principal do IAM com permissões para `create` e `describe` um cluster do Amazon EKS. Para ter mais informações, consulte Criar um cluster local do Kubernetes em um Outpost e Listar ou descrever todos os clusters.

Para criar um cluster do Amazon EKS.

1. Se já tiver um perfil do IAM de cluster ou se for criar seu cluster com `eksctl`, você poderá ignorar essa etapa. Por padrão, `eksctl` cria um perfil para você.

   Para criar um perfil do IAM de cluster do Amazon EKS

   1. Execute o seguinte comando para criar um arquivo JSON de política de confiança do IAM:

      ```
      cat >eks-cluster-role-trust-policy.json <<EOF
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "eks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      }
      EOF
      ```

   2. Crie o perfil do IAM do cluster do Amazon EKS. Se necessário, prefixe *eks-cluster-role-trust-policy.json* com o caminho no computador no qual você gravou o arquivo na etapa anterior. O comando associa a política de confiança criada na etapa anterior à função. Para criar um perfil do IAM, a entidade principal do IAM que estiver criando o perfil deverá ser atribuída à seguinte ação `iam:CreateRole` (permissão):

      ```
      aws iam create-role --role-name myAmazonEKSClusterRole --assume-role-policy-
      document file://"eks-cluster-role-trust-policy.json"
      ```

   3. Você pode atribuir a política gerenciada do Amazon EKS ou criar sua própria política personalizada. Para obter as permissões mínimas que você deve usar em sua política personalizada, consulte Função do IAM do cluster do Amazon EKS.

      Anexe a política gerenciada do Amazon EKS denominada `AmazonEKSClusterPolicy` à função. Para anexar uma política do IAM a uma entidade principal do IAM a entidade principal do IAM que está anexando a política deve receber uma das seguintes ações do IAM (permissões): `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name myAmazonEKSClusterRole
```

2. Crie um cluster do Amazon EKS.

   É possível criar um cluster usando `eksctl`, o AWS Management Console ou a AWS CLI.

   eksctl

   Pré-requisito

   Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

   Para criar um cluster

   Crie um cluster `IPv4` do Amazon EKS com a versão padrão do Kubernetes do Amazon EKS na sua Região da AWS padrão. Antes da execução do comando, realize as seguintes substituições:

   - Substitua *`region-code`* pela Região da AWS na qual você deseja criar o cluster.
   - Substitua *`my-cluster`* por um nome de cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
   - Substitua *`1.29`* por qualquer versão compatível do Amazon EKS.

     > **Note**
     >
     > Para implantar em um cluster 1.30 no momento, você precisa usar o AWS Management Console ou a AWS CLI.

   - Altere os valores para `vpc-private-subnets` para atender às suas necessidades. Também é possível adicionar outras IDs. Você deve especificar pelo menos dois IDs de sub-rede. Se preferir especificar sub-redes públicas, poderá alterar `--vpc-private-subnets` para `--vpc-public-subnets`. Sub-redes públicas têm uma tabela de rotas

associada a uma rota para um gateway da Internet, mas sub-redes privadas não têm tabelas de rotas associadas. Convém utilizar sub-redes privadas sempre que possível.

As sub-redes escolhidas devem atender aos Requisitos para sub-redes do Amazon EKS. Antes de selecionar sub-redes, convém estar familiarizado com todos os Requisitos e considerações sobre VPCs e sub-redes do Amazon EKS.

```
eksctl create cluster --name my-cluster --region region-code --version 1.29 --
vpc-private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
```

O provisionamento de cluster leva alguns minutos. Durante a criação do cluster, várias linhas de saída são exibidas. A última linha do resultado é semelhante ao exemplo de linha a seguir.

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> Para ver a maioria das opções que podem ser especificadas ao criar um cluster com `eksctl`, use o comando **`eksctl create cluster --help`**. Para ver todas as opções disponíveis, é possível utilizar um arquivo `config`. Para obter mais informações, consulte Uso dos arquivos de configuração e o esquema de arquivo de configuração na documentação do `eksctl`. Você pode encontrar exemplos de arquivos de configuração no GitHub.

Configurações opcionais

Veja a seguir as configurações opcionais que, se necessário, devem ser adicionadas ao comando anterior. Apenas é possível habilitar essas opções ao criar o cluster, e não depois. Se precisar especificar essas opções, será necessário criar o cluster com um arquivo de configuração `eksctl` e especificar as configurações em vez de utilizar o comando anterior.

- Para especificar um ou mais grupos de segurança que o Amazon EKS atribui às interfaces de rede que ele cria, especifique a opção `securityGroup`.

Independentemente de você escolher algum grupo de segurança, o Amazon EKS cria um grupo de segurança que permite comunicação entre seu cluster e sua VPC. O Amazon EKS associa esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte the section called “Requisitos para grupos de segurança”. É possível modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS.

- Se quiser especificar a qual bloco de Encaminhamento entre Domínios Sem Classificação (CIDR) `IPv4` o Kubernetes atribui endereços IP de serviço, especifique a opção `serviceIPv4CIDR`.

  Especificar seu próprio intervalo pode ajudar a evitar conflitos entre serviços Kubernetes e outras redes com emparelhamento ou conectadas à sua VPC. Insira um intervalo em notação CIDR. Por exemplo: `10.2.0.0/16`.

  O bloco CIDR deve atender aos seguintes requisitos:

  - Estar dentro de um dos seguintes intervalos: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.
  - Ter um tamanho mínimo de `/24` e máximo de `/12`.
  - Não se sobrepor ao intervalo da VPC para seus recursos do Amazon EKS.

  Apenas é possível especificar essa opção ao utilizar a família de endereços `IPv4` e somente ao criar o cluster. Se isso não for especificado, o Kubernetes atribuirá endereços IP de serviço de qualquer um dos blocos CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Se estiver criando um cluster e quiser que esse cluster atribua endereços `IPv6` a Pods e serviços em vez de endereços `IPv4`, especifique a opção `ipFamily`.

  Por padrão, o Kubernetes atribui endereços `IPv4` a Pods e serviços. Antes de optar por utilizar a família `IPv6`, familiarize-se com todas as considerações e requisitos nos tópicos the section called “Requisitos e considerações para VPCs”, the section called “Requisitos e considerações para sub-redes”, the section called “Requisitos para grupos de segurança” e the section called “IPv6”. Se você escolher a família `IPv6`, não poderá especificar um intervalo de endereços para o Kubernetes atribuir endereços de serviço `IPv6` da mesma forma que pode ser feita para a família `IPv4`. O Kubernetes atribui endereços de serviço com base no intervalo de endereços local exclusivo (`fc00::/7`).

AWS Management Console

Para criar um cluster

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha Add cluster (Adicionar cluster) e, em seguida, Create (Criar).
3. Na página Configure cluster (Configurar cluster), preencha os seguintes campos:
   - Name (Nome): um nome exclusivo para o cluster. O nome pode conter apenas caracteres alfanuméricos (diferencia maiúsculas de minúsculas), hifens e sublinhados. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
   - Versão do Kubernetes: a versão do Kubernetes a ser usada no cluster. Recomendamos que você selecione a versão mais recente, a menos que precise de uma versão anterior.
   - Tipo de suporte: a política de versão do Kubernetes que você gostaria de definir para seu cluster. Se desejar que seu cluster seja executado somente em uma versão com suporte padrão, escolher Suporte padrão é uma opção possível. Se desejar que seu cluster entre no suporte estendido ao final do suporte padrão para uma versão, escolha Suporte estendido. Se você selecionar uma versão do Kubernetes que esteja atualmente em suporte estendido, não poderá selecionar o suporte padrão como opção.
   - Cluster Service Role (Perfil de serviço de cluster): escolha o perfil do IAM do cluster do Amazon EKS que você criou para permitir que o ambiente de gerenciamento do Kubernetes gerencie os recursos da AWS em seu nome.
   - Secrets encryption (Criptografia de segredos): (opcional) escolha habilitar a criptografia de segredos do Kubernetes usando uma chave do KMS. Também é possível isso depois de criar o cluster. Antes de habilitar esse recurso, familiarize-se com as informações em Criptografar segredos do Kubernetes com o AWS KMS em clusters existentes.
   - Tags (Etiquetas) – (opcional) adicione etiquetas ao cluster. Para obter mais informações, consulte Organizar recursos do Amazon EKS com tags.

     Quando terminar com essa página, escolha Próximo.

4. Na página Specify networking (Especificar redes), selecione valores para os seguintes campos:

   - VPC: escolha uma VPC existente que atenda aos Requisitos de VPC do Amazon EKS na qual criar o cluster. Antes de escolher uma VPC, convém familiarizar-se com todos os requisitos e considerações em Requisitos e considerações sobre a VPC e a sub-rede do Amazon EKS. Não será possível alterar quais VPCs você deseja utilizar depois de criar o cluster. Se nenhuma VPC estiver listada, você precisará criar uma primeiro. Para obter mais informações, consulte Criar uma VPC para o cluster do Amazon EKS.
   - Subnets (Sub-redes): por padrão, as sub-redes disponíveis na VPC especificada no campo anterior são pré-selecionadas. É necessário selecionar pelo menos duas.

     As sub-redes escolhidas devem atender aos Requisitos para sub-redes do Amazon EKS. Antes de selecionar sub-redes, convém estar familiarizado com todos os Requisitos e considerações sobre VPCs e sub-redes do Amazon EKS.

     Security groups (Grupos de segurança) (Opcional): especifique um ou mais grupos de segurança que você deseja que o Amazon EKS associe às interfaces de rede que ele cria.

     Independentemente de você escolher algum grupo de segurança, o Amazon EKS cria um grupo de segurança que permite comunicação entre seu cluster e sua VPC. O Amazon EKS associa esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte the section called “Requisitos para grupos de segurança”. É possível modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS.
   - Escolher família de endereços IP do cluster: é possível escolher IPv4 e IPv6.

     Por padrão, o Kubernetes atribui endereços `IPv4` a Pods e serviços. Antes de optar por utilizar a família `IPv6`, familiarize-se com todas as considerações e requisitos nos tópicos the section called “Requisitos e considerações para VPCs”, the section called “Requisitos e considerações para sub-redes”, the section called “Requisitos para grupos de segurança” e the section called “IPv6”. Se você escolher a família `IPv6`, não poderá especificar um intervalo de endereços para o Kubernetes atribuir endereços de serviço `IPv6` da mesma forma que pode ser feita para a família `IPv4`. O Kubernetes atribui endereços de serviço com base no intervalo de endereços local exclusivo (`fc00::/7`).

- (Opcional) Escolha Configurar intervalo de endereços IP do serviço Kubernetes e especifique um Intervalo de serviço do **IPv4**.

  Especificar seu próprio intervalo pode ajudar a evitar conflitos entre serviços Kubernetes e outras redes com emparelhamento ou conectadas à sua VPC. Insira um intervalo em notação CIDR. Por exemplo: `10.2.0.0/16`.

  O bloco CIDR deve atender aos seguintes requisitos:

  - Estar dentro de um dos seguintes intervalos: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.
  - Ter um tamanho mínimo de `/24` e máximo de `/12`.
  - Não se sobrepor ao intervalo da VPC para seus recursos do Amazon EKS.

  Apenas é possível especificar essa opção ao utilizar a família de endereços `IPv4` e somente ao criar o cluster. Se isso não for especificado, o Kubernetes atribuirá endereços IP de serviço de qualquer um dos blocos CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Para Cluster endpoint access (Acesso ao endpoint do cluster), selecione uma opção. Depois de criar o cluster, você poderá alterar essa opção. Antes de selecionar uma opção não padrão, familiarize-se com as opções e suas implicações. Para obter mais informações, consulte Controlar o acesso à rede ao endpoint do servidor de API do cluster.

  Quando terminar com essa página, escolha Próximo.

5. (Opcional) Na página Configurar observabilidade, escolha quais opções de Métricas e Log do ambiente de gerenciamento deseja ativar. Por padrão, o cada tipo de log está desativado.

   - Para obter mais informações sobre as opções de métricas do Prometheus, consulte Etapa 1: ativar as métricas do Prometheus ao criar um cluster.
   - Para obter mais informações sobre as opções do Log do ambiente de gerenciamento, consulte Enviar logs do ambiente de gerenciamento para o CloudWatch Logs.

   Quando terminar com essa página, escolha Próximo.

6. Na página Select add-ons (Selecionar complementos), escolha os complementos que você deseja adicionar ao cluster. Você pode escolher tantos complementos do Amazon EKS e complementos do AWS Marketplace quantos forem necessários. Se os

complementos do AWS Marketplace que você quer instalar não estiverem listados, será possível pesquisar os complementos do AWS Marketplace disponíveis inserindo texto na caixa de pesquisa. Você também poderá pesquisar por category (categoria), vendor (fornecedor) ou pricing model (modelo de preços) e depois escolher os complementos nos resultados da pesquisa. Quando terminar com essa página, escolha Próximo.

Alguns complementos, como Amazon VPC CNI, CoreDNS e kube-proxy, são instalados por padrão. Se você desabilitar qualquer um dos complementos padrão, isso poderá afetar sua capacidade de executar aplicações do Kubernetes.

7. Na página Configurar opções de complementos selecionados, selecione a versão que deseja instalar. Você sempre pode atualizar para uma versão posterior após a criação do cluster. Você pode atualizar a configuração de cada complemento após a criação do cluster. Para obter mais informações sobre a configuração de complementos, consulte Atualizar um complemento do Amazon EKS. Após terminar com essa página, escolha Próximo.
8. Na página Review and create (Revisar e criar), revise as informações que você inseriu ou selecionou nas páginas anteriores. Se precisar fazer alterações, escolha Edit (Editar). Quando estiver satisfeito, escolha Create (Criar). O campo Status mostra o campo CREATING (Criando) enquanto o cluster é provisionado.

> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte Insufficient capacity (Capacidade insuficiente).

O provisionamento de cluster leva alguns minutos.

AWS CLI

Para criar um cluster

1. Crie o cluster usando o comando a seguir. Antes da execução do comando, realize as seguintes substituições:
   - Substitua *region-code* pela Região da AWS na qual você deseja criar o cluster.
   - Substitua *my-cluster* por um nome de cluster. O nome pode conter apenas caracteres alfanuméricos (diferencia maiúsculas de minúsculas), hifens e sublinhados. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
   - Substitua *1.30* por qualquer versão compatível do Amazon EKS.
   - Substitua *111122223333* pelo ID da conta e *myAmazonEKSClusterRole* pelo nome do perfil do IAM do cluster.
   - Substitua os valores de `subnetIds` pelos seus próprios valores. Também é possível adicionar outras IDs. Você deve especificar pelo menos dois IDs de sub-rede.

     As sub-redes escolhidas devem atender aos Requisitos para sub-redes do Amazon EKS. Antes de selecionar sub-redes, convém estar familiarizado com todos os Requisitos e considerações sobre VPCs e sub-redes do Amazon EKS.
   - Se não quiser especificar um ID de grupo de segurança, remova `,securityGroupIds=sg-`*ExampleID1* do comando. Se quiser especificar um ou mais IDs de grupo de segurança, substitua os valores de `securityGroupIds` pelos seus próprios. Também é possível adicionar outras IDs.

     Independentemente de você escolher algum grupo de segurança, o Amazon EKS cria um grupo de segurança que permite comunicação entre seu cluster e sua VPC. O Amazon EKS associa esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte the section called “Requisitos para grupos de segurança”. É possível modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS.

```
aws eks create-cluster --region region-code --name my-cluster --kubernetes-
version 1.30 \
```

```
  --role-arn arn:aws:iam::111122223333:role/myAmazonEKSClusterRole \
  --resources-vpc-config
subnetIds=subnet-ExampleID1,subnet-ExampleID2,securityGroupIds=sg-ExampleID1
```

> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte Insufficient capacity (Capacidade insuficiente).

Configurações opcionais

Veja a seguir as configurações opcionais que, se necessário, devem ser adicionadas ao comando anterior. Apenas é possível habilitar essas opções ao criar o cluster, e não depois.

- Por padrão, o EKS instala vários complementos de rede durante a criação do cluster. Isso inclui o Amazon VPC CNI, o CoreDNS e o kube-proxy.

  Se você quiser desabilitar a instalação desses complementos de rede padrão, use o parâmetro abaixo. Essa opção poderá ser usada para CNIs alternativos, como Cilium. Para obter mais informações, confira a API Reference do EKS.

  ```
  aws eks create-cluster --bootstrapSelfManagedAddons false
  ```

- Se quiser especificar de qual bloco de Encaminhamento Entre Domínios Sem Classificação (CIDR) `IPv4` o Kubernetes atribui endereços IP de serviço, você deve especificá-lo adicionando **`--kubernetes-network-config serviceIpv4Cidr=`*`CIDR block`*** ao comando a seguir.

  Especificar seu próprio intervalo pode ajudar a evitar conflitos entre serviços Kubernetes e outras redes com emparelhamento ou conectadas à sua VPC. Insira um intervalo em notação CIDR. Por exemplo: `10.2.0.0/16`.

  O bloco CIDR deve atender aos seguintes requisitos:

- Estar dentro de um dos seguintes intervalos: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.
- Ter um tamanho mínimo de `/24` e máximo de `/12`.
- Não se sobrepor ao intervalo da VPC para seus recursos do Amazon EKS.

Apenas é possível especificar essa opção ao utilizar a família de endereços `IPv4` e somente ao criar o cluster. Se isso não for especificado, o Kubernetes atribuirá endereços IP de serviço de qualquer um dos blocos CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Se estiver criando um cluster e quiser que o cluster atribua endereços `IPv6` a Pods e serviços em vez de endereços `IPv4`, adicione **`--kubernetes-network-config ipFamily=ipv6`** ao comando a seguir.

  Por padrão, o Kubernetes atribui endereços `IPv4` a Pods e serviços. Antes de optar por utilizar a família `IPv6`, familiarize-se com todas as considerações e requisitos nos tópicos the section called “Requisitos e considerações para VPCs”, the section called “Requisitos e considerações para sub-redes”, the section called “Requisitos para grupos de segurança” e the section called “IPv6”. Se você escolher a família `IPv6`, não poderá especificar um intervalo de endereços para o Kubernetes atribuir endereços de serviço `IPv6` da mesma forma que pode ser feita para a família `IPv4`. O Kubernetes atribui endereços de serviço com base no intervalo de endereços local exclusivo (`fc00::/7`).

2. Leva alguns minutos para provisionar o cluster. Você pode consultar o status do cluster com o comando a seguir.

   ```
   aws eks describe-cluster --region region-code --name my-cluster --query
    "cluster.status"
   ```

   Não prossiga para a próxima etapa até que a saída recebida esteja `ACTIVE`.

3. Se você criou seu cluster usando `eksctl`, pode ignorar esta etapa. Isso ocorre porque `eksctl` já concluiu essa etapa para você. Habilite o `kubectl` para se comunicar com o cluster adicionando um novo contexto ao arquivo `kubectl config`. Para obter mais informações sobre como criar e atualizar o arquivo, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig.

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

Veja um exemplo de saída abaixo.

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/username/.kube/config
```

4. Confirme a comunicação com o cluster, executando o seguinte comando:

```
kubectl get svc
```

Veja um exemplo de saída abaixo.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

5. (Recomendado) Para usar alguns complementos do Amazon EKS ou para permitir que workloads individuais do Kubernetes tenham permissões específicas do AWS Identity and Access Management (IAM), crie um provedor OpenID Connect (OIDC) do IAM para o cluster. Você só precisa criar um provedor de OIDC do IAM para o cluster uma vez. Para saber mais sobre os complementos do Amazon EKS, consulte Complementos do Amazon EKS. Para saber mais sobre a atribuição de permissões específicas do IAM às workloads, consulte Perfis do IAM para contas de serviço.
6. (Recomendado) Configure o cluster para o plugin Amazon VPC CNI plugin for Kubernetes antes de implantar nós do Amazon EC2 no cluster. Por padrão, o plugin foi instalado com o seu cluster. Quando nós do Amazon EC2 são adicionados ao cluster, o plugin é implantado automaticamente em cada nó do Amazon EC2 que você adicionar. O plugin requer a anexação de uma das seguintes políticas do IAM a um perfil do IAM:

   Política do IAM gerenciada da **`AmazonEKS_CNI_Policy`**

   Se o cluster usar a família `IPv4`

   Uma política do IAM que você cria

   Se o cluster usar a família `IPv6`

   O perfil do IAM ao qual a política é anexada pode ser o perfil do IAM do nó ou um perfil dedicado utilizado apenas para o plugin. Convém anexar a política a esse perfil. Para obter mais informações sobre como criar a função, consulte Configuração do Amazon VPC CNI plugin for

Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA) ou Perfil do IAM em nós do Amazon EKS.

7. Se você implantou seu cluster usando o AWS Management Console, pode ignorar esta etapa. O AWS Management Console implanta os complementos Amazon VPC CNI plugin for Kubernetes, CoreDNS e `kube-proxy` do Amazon EKS por padrão.

   (Opcional) Se você implantar o cluster usando o `eksctl` ou a AWS CLI, os complementos Amazon VPC CNI plugin for Kubernetes, CoreDNS e `kube-proxy` autogerenciados serão implantados. É possível migrar os complementos autogerenciados Amazon VPC CNI plugin for Kubernetes, CoreDNS e `kube-proxy`, que são implantados com seu cluster, para complementos do Amazon EKS. Para obter mais informações, consulte Complementos do Amazon EKS.
8. (Opcional) Se você ainda não fez isso, poderá habilitar as métricas do Prometheus para seu cluster. Para obter mais informações, consulte Criar um extrator no Guia do usuário do Amazon Managed Service for Prometheus.
9. Se você habilitou as métricas do Prometheus, deverá configurar seu `aws-auth ConfigMap` para conceder ao extrator permissões no cluster. Para obter mais informações, consulte Configurar o cluster do Amazon EKS no Guia do usuário do Amazon Managed Service for Prometheus.
10. Se você planeja implantar workloads em seu cluster que usam volumes do Amazon EBS e tiver criado um cluster `1.23` ou posterior, deverá instalar o Armazene volumes do Kubernetes com o Amazon EBS em seu cluster antes de implantar as workloads.

Próximas etapas recomendadas:

- A entidade principal do IAM que criou o cluster é a única que tem acesso ao cluster. Conceda permissões a outras entidades principais do IAM para que elas possam acessar o cluster.
- Se a entidade principal do IAM que criou o cluster tiver apenas as permissões mínimas do IAM referenciadas nos pré-requisitos, então talvez seja interessante adicionar mais permissões do Amazon EKS a essa entidade principal. Para obter mais informações sobre como conceder permissões do Amazon EKS a entidades principais do IAM, consulte Identity and Access Management para o Amazon EKS.
- Se você quiser que as entidades principais do IAM que criaram o cluster, ou qualquer outra entidade principal do IAM, visualize os recursos do Kubernetes no console do Amazon EKS, conceda a elas Permissões obrigatórias.

- Se quiser que os nós e as entidades principais do IAM acessem o cluster de dentro da VPC, habilite o endpoint privado do seu cluster. Por padrão, o endpoint público está habilitado. É possível desativar o endpoint público depois de habilitar o endpoint privado, se desejado. Para obter mais informações, consulte Controlar o acesso à rede ao endpoint do servidor de API do cluster.
- Habilitar a criptografia de segredos para o seu cluster.
- Configurar o registro em log para o seu cluster.
- Adicionar nós ao seu cluster.

Prepare-se para atualizações de versão do Kubernetes com insights de cluster

Os insights de cluster do Amazon EKS fornecem recomendações para ajudar você a seguir as práticas recomendadas do Amazon EKS e do Kubernetes. Cada cluster do Amazon EKS passa por verificações automáticas e recorrentes em relação a uma lista de insights com curadoria do Amazon EKS. Essas verificações de insights são totalmente gerenciadas pelo Amazon EKS e oferecem recomendações sobre como lidar com quaisquer descobertas.

Uso recomendado de insights de cluster:

- Antes de atualizar a versão do cluster do Kubernetes, verifique os insights do cluster no console do EKS.
- Se seu cluster identificou problemas, revise-os e faça as correções apropriadas. Os problemas incluem links para o Amazon EKS e o Kubernetes.
- Depois de corrigir os problemas, aguarde a atualização dos insights do cluster. Se todos os problemas tiverem sido resolvidos, atualize seu cluster.

Atualmente, o Amazon EKS retorna apenas informações relacionadas à prontidão para atualização da versão do Kubernetes.

Os insights de atualização identificam possíveis problemas que podem afetar os upgrades do cluster do Kubernetes. Isso minimiza o esforço que os administradores gastam na preparação para atualizações e aumenta a confiabilidade dos aplicativos nas versões mais recentes do Kubernetes. Os clusters são verificados automaticamente pelo Amazon EKS em relação a uma lista de possíveis problemas que afetam a atualização da versão do Kubernetes. O Amazon EKS atualiza

frequentemente a lista de verificações de insights com base nas análises das alterações feitas em cada versão do Kubernetes.

Os insights de atualização do Amazon EKS aceleram o processo de teste e verificação de novas versões. Eles também permitem que administradores de clusters e desenvolvedores de aplicações aproveitem os recursos mais recentes do Kubernetes destacando preocupações e oferecendo conselhos sobre remediação. Para ver a lista de verificações de insights realizadas e quaisquer problemas relevantes identificados pelo Amazon EKS, você pode chamar a operação `ListInsights` da API do Amazon EKS ou consultar o console do Amazon EKS.

Os insights do cluster são atualizados periodicamente. Você não pode atualizar manualmente os insights do cluster. Se você corrigir um problema de cluster, levará algum tempo para que os insights do cluster sejam atualizados. Para determinar se uma correção foi bem-sucedida, compare a hora em que a alteração foi implantada com a "hora da última atualização" do insight do cluster.

Visualizar insights do cluster (console)

Para visualizar insights de um cluster do Amazon EKS:

a. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

b. Na lista Clusters, selecione o nome do cluster do Amazon EK cujos insights deseja visualizar.

c. Escolha a guia Atualizar insights.

d. Na página Atualizar insights, você verá os seguintes campos:

- Nome: a verificação que foi realizada pelo Amazon EKS em relação ao cluster.
- Status do insight: um insight com status de “Erro” normalmente significa que a versão do Kubernetes afetada é N+1 da versão atual do cluster, enquanto um status de “Aviso” significa que o insight se aplica a uma versão do Kubernetes futura N+2 ou mais. Um insight com status de “Aprovado” significa que o Amazon EKS não encontrou nenhum problema associado a essa verificação de insights em seu cluster. Um status de insight de “Desconhecido” significa que o Amazon EKS não consegue determinar se seu cluster é afetado por essa verificação de insights.
- Versão: a versão do Kubernetes que o insight verificou em busca de possíveis problemas.
- Horário da última atualização (UTC- 5:00): a hora em que o status do insight foi atualizado pela última vez para esse cluster.
- Horário da última transição (UTC- 5:00): a hora em que o status desse insight foi alterado pela última vez.

- Descrição: informações da verificação de insights, que incluem o alerta e as ações recomendadas para remediação.

Visualizar insights do cluster (AWS CLI)

Para visualizar insights de um cluster do Amazon EKS:

a. Determine qual cluster você gostaria de verificar para obter informações. O comando a seguir lista insights de um cluster especificado. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado:

- Substitua *region-code* pelo código da sua Região da AWS.
- Substitua o *my-cluster* pelo nome do cluster.

```
aws eks list-insights --region region-code --cluster-name my-cluster
```

Veja um exemplo de saída abaixo.


```
{
    "insights": [
        {
            "category": "UPGRADE_READINESS",
            "name": "Deprecated APIs removed in Kubernetes v1.29",
            "insightStatus": {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30 days."
            },
            "kubernetesVersion": "1.29",
            "lastTransitionTime": 1698774710.0,
            "lastRefreshTime": 1700157422.0,
            "id": "123e4567-e89b-42d3-a456-579642341238",
            "description": "Checks for usage of deprecated APIs that are scheduled
 for removal in Kubernetes v1.29. Upgrading your cluster before migrating to the
 updated APIs supported by v1.29 could cause application impact."
        }
    ]
}
```


b. Para obter informações descritivas sobre o insight, execute o comando a seguir. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado:

- Substitua *region-code* pelo código da sua Região da AWS.
- Substitua *123e4567-e89b-42d3-a456-579642341238* pelo ID do insight obtido da listagem dos insights do cluster.
- Substitua o *my-cluster* pelo nome do cluster.

```
aws eks describe-insight --region region-code --id 123e4567-e89b-42d3-
a456-579642341238 --cluster-name my-cluster
```

Veja um exemplo de saída abaixo.


```
{
    "insight": {
        "category": "UPGRADE_READINESS",
        "additionalInfo": {
            "EKS update cluster documentation": "https://docs.aws.amazon.com/eks/
latest/userguide/update-cluster.html",
            "Kubernetes v1.29 deprecation guide": "https://kubernetes.io/docs/
reference/using-api/deprecation-guide/#v1-29"
        },
        "name": "Deprecated APIs removed in Kubernetes v1.29",
        "insightStatus": {
            "status": "PASSING",
            "reason": "No deprecated API usage detected within the last 30 days."
        },
        "kubernetesVersion": "1.29",
        "recommendation": "Update manifests and API clients to use newer Kubernetes
 APIs if applicable before upgrading to Kubernetes v1.29.",
        "lastTransitionTime": 1698774710.0,
        "lastRefreshTime": 1700157422.0,
        "categorySpecificSummary": {
            "deprecationDetails": [
                {
                    "usage": "/apis/flowcontrol.apiserver.k8s.io/v1beta2/
flowschemas",
                    "replacedWith": "/apis/flowcontrol.apiserver.k8s.io/v1beta3/
flowschemas",
                    "stopServingVersion": "1.29",
                    "clientStats": [],
                    "startServingReplacementVersion": "1.26"
                },
                {
```

```
                    "usage": "/apis/flowcontrol.apiserver.k8s.io/v1beta2/
prioritylevelconfigurations",
                    "replacedWith": "/apis/flowcontrol.apiserver.k8s.io/v1beta3/
prioritylevelconfigurations",
                    "stopServingVersion": "1.29",
                    "clientStats": [],
                    "startServingReplacementVersion": "1.26"
                }
            ]
        },
        "id": "f6a11fe4-77f7-48c6-8326-9a13f022ecb3",
        "resources": [],
        "description": "Checks for usage of deprecated APIs that are scheduled for
 removal in Kubernetes v1.29. Upgrading your cluster before migrating to the updated
 APIs supported by v1.29 could cause application impact."
    }
}
```


Atualizar um cluster existente para a nova versão do Kubernetes

Quando uma nova versão do Kubernetes está disponível no Amazon EKS, você pode atualizar o cluster do Amazon EKS para a versão mais recente.

> **Important**
>
> Depois de atualizar um cluster, não é possível revertê-lo uma versão anterior. Antes de atualizar para uma nova versão do Kubernetes, recomendamos revisar as informações em Compreeender o ciclo de vida da versão do Kubernetes no EKS e também as etapas de atualização neste tópico.

As novas versões do Kubernetes às vezes introduzem alterações significativas. Portanto, recomendamos que você teste o comportamento das aplicações com relação a uma nova versão do Kubernetes antes de atualizar os clusters de produção. Você pode fazer isso, criando um fluxo de trabalho de integração contínua para testar o comportamento da aplicação antes de mudar para uma nova versão do Kubernetes.

O processo de atualização consiste no Amazon EKS iniciar novos nós do servidor de API com a versão do Kubernetes atualizada para substituir os nós existentes. O Amazon EKS executa verificações padrão de integridade e prontidão da infraestrutura para o tráfego de rede nesses

novos nós para confirmar que eles estão funcionando conforme o esperado. Porém, depois que você iniciou a atualização do cluster, não pode pausá-la nem interrompê-la. Se qualquer uma dessas verificações falhar, o Amazon EKS reverterá a implantação de infraestrutura e o cluster permanecerá na versão anterior do Kubernetes. A execução de aplicações não será afetada e o cluster nunca será deixado em um estado irrecuperável ou não determinista. O Amazon EKS faz backup regularmente de todos os clusters gerenciados, além de ter mecanismos para recuperar clusters, se necessário. Estamos avaliando e melhorando constantemente nossos processos de gerenciamento de infraestrutura do Kubernetes.

Para atualizar o cluster, o Amazon EKS requer até cinco endereços IP disponíveis das sub-redes que você especificou ao criar o cluster. O Amazon EKS cria novas interfaces de rede elásticas (interfaces de rede) de cluster em qualquer uma das sub-redes especificadas. Como as interfaces de rede podem ser criadas em sub-redes diferentes das interfaces de rede existentes, certifique-se de que as regras do grupo de segurança permitam a comunicação de cluster necessária para qualquer uma das sub-redes que você especificou ao criar o cluster. Se qualquer uma das sub-redes especificadas quando você criou o cluster não existir, não tiver endereços IP suficientes disponíveis ou não tiver regras de grupo de segurança que permitam a comunicação de cluster necessária, a atualização poderá falhar.

> **Note**
>
> Para garantir que o endpoint do servidor de API do seu cluster sempre esteja acessível, o Amazon EKS fornece um ambiente de gerenciamento Kubernetes com alta disponibilidade e executa atualizações contínuas das instâncias do servidor de API durante as operações de atualização. Para contabilizar a alteração de endereços IP das instâncias do servidor de API que oferecem suporte ao seu endpoint do servidor de API Kubernetes, você deve garantir que seus clientes do servidor de API gerenciem as reconexões de modo eficaz. Versões recentes do `kubectl` e das bibliotecas de cliente Kubernetes que têm suporte oficial executam esse processo de reconexão de modo transparente.

Atualizar a versão do Kubernetes de um cluster do Amazon EKS

Para atualizar a versão do Kubernetes para o cluster.

1. Compare a versão do Kubernetes do ambiente de gerenciamento do cluster com a versão do Kubernetes dos nós.

- Obtenha a versão do Kubernetes para seu ambiente de gerenciamento do cluster.

```
kubectl version
```

- Obtenha a versão do Kubernetes para seus nós. Esse comando retorna todos os nós autogerenciados e gerenciados do Amazon EC2 e do Fargate. Cada Pod do Fargate é listado como seu próprio nó.

```
kubectl get nodes
```

Antes de atualizar o ambiente de gerenciamento para uma nova versão do Kubernetes, certifique-se de que a versão secundária do Kubernetes dos nós gerenciados e dos nós do Fargate no cluster sejam iguais à versão do ambiente de gerenciamento. Por exemplo, se o ambiente de gerenciamento estiver executando a versão `1.29` e um dos nós estiver executando a versão `1.28`, será necessário atualizar os nós para a versão `1.29` antes de atualizar o ambiente de gerenciamento para 1.30. Também recomendamos que você atualize seus nós autogerenciados para a mesma versão do plano de controle antes de atualizar o plano de controle. Para ter mais informações, consulte Atualizar um grupo de nós gerenciados para seu cluster e Atualizar nós autogerenciados para seu cluster. Se você tiver nós do Fargate com uma versão secundária inferior à versão do ambiente de gerenciamento, primeiramente exclua o Pod que é representado pelo nó. Em seguida, atualize seu ambiente de gerenciamento. Todos os demais Pods serão atualizados para a nova versão depois de reimplantados.

2. Se a versão do Kubernetes com a qual você implantou originalmente o cluster foi a versão `1.25` ou posterior do Kubernetes, pule essa etapa.

   Por padrão, o controlador de admissão de política de segurança de Pod é habilitado nos clusters do Amazon EKS. Antes de atualizar o cluster, certifique-se de que as políticas de segurança de Pod adequadas estejam em vigor. Isso acontece para evitar possíveis problemas de segurança. Você pode verificar a política padrão com o comando **`kubectl get psp eks.privileged`**.

   ```
   kubectl get psp eks.privileged
   ```

   Se receber o seguinte erro, consulte Política de segurança de Pod padrão do Amazon EKS antes de prosseguir.

```
Error from server (NotFound): podsecuritypolicies.extensions "eks.privileged" not
 found
```

3. Se a versão do Kubernetes com a qual você implantou originalmente o cluster foi a versão `1.18` ou posterior do Kubernetes, pule essa etapa.

   Talvez seja necessário remover um termo descontinuado do seu manifesto CoreDNS.

   a. Verifique se o manifesto do CoreDNS tem uma linha que tenha apenas a palavra `upstream`.

   ```
   kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
    grep upstream
   ```

   Se nenhuma saída for retornada, significa que o manifesto não tem a linha. Se este for o caso, pule para a próxima etapa. Se a palavra `upstream` for retornada, remova a linha.

   b. Remova a linha perto do topo do arquivo que só tem a palavra `upstream` no arquivo de configuração. Não mude mais nada no arquivo. Depois que a linha for removida, salve as alterações.

   ```
   kubectl edit configmap coredns -n kube-system -o yaml
   ```

4. Atualize seu cluster usando `eksctl`, o AWS Management Console ou a AWS CLI.

> **Important**
>
> - Se você estiver atualizando para a versão `1.23` e usar volumes do Amazon EBS em seu cluster, será necessário instalar o driver de CSI do Amazon EBS no cluster antes de atualizar o cluster para a versão `1.23` a fim de evitar interrupções de workload. Para ter mais informações, consulte Kubernetes 1.23 e Armazene volumes do Kubernetes com o Amazon EBS.
> - O Kubernetes `1.24` e versões posteriores são usados `containerd` como o runtime padrão do contêiner. Se você estiver migrando para o `containerd` runtime e já tiver Fluentd configurado paraContainer Insights, deverá migrar Fluentd para ele Fluent Bit antes de atualizar seu cluster. Os analisadores Fluentd são configurados para analisar apenas mensagens de log no formato JSON. Ao contrário de `dockerd`, o runtime do `containerd` contêiner tem mensagens de log que não estão no formato JSON.

Se você não migrar para Fluent Bit, alguns dos Fluentd's analisadores configurados gerarão muitos erros dentro do Fluentd contêiner. Para obter mais informações sobre migração, consulte Configurar Fluent Bit como um DaemonSet para enviar logs para o CloudWatch Logs.

- Como o Amazon EKS executa um plano de controle de alta disponibilidade, você pode atualizar apenas uma versão secundária por vez. Para obter mais informações sobre esse requisito, consulte Política de suporte a versões e diferenciação de versões do Kubernetes. Suponha que a versão atual do cluster seja a `1.28` e você queira atualizar para a versão `1.30`. Você deve primeiro atualizar o cluster versão `1.28` para versão `1.29` e depois atualizar o cluster versão `1.29` para a versão `1.30`.
- Analise a distorção de versão entre o `kube-apiserver` e o `kubelet` do Kubernetes em seus nós.
  - Com a versão `1.28` do Kubernetes e posteriores, o `kubelet` pode ter até três versões secundárias anteriores ao `kube-apiserver`. Consulte a política de distorção de versão upstream do Kubernetes.
  - Se o `kubelet` em seus nós gerenciados e nós do Fargate estiver em na versão `1.25` ou posterior do Kubernetes, você poderá atualizar seu cluster com até três versões à frente sem atualizar a versão do `kubelet`. Por exemplo, se a versão `1.25` do `kubelet` estiver ativa, você poderá atualizar a versão do cluster do Amazon EKS de `1.25` para `1.26` e de `1.27` para `1.28`, enquanto o `kubelet` estiver na versão `1.25`.
  - Se o `kubelet` nos nós gerenciados e do Fargate estiver em uma versão `1.24` ou anterior do Kubernetes, pode ser que haja apenas duas versões secundárias anteriores ao `kube-apiserver`. Em outras palavras, se o `kubelet` estiver na versão `1.24` ou anterior, você só poderá atualizar seu cluster até duas versões à frente. Por exemplo, se o `kubelet` estiver na versão `1.21`, você poderá atualizar a versão do cluster do Amazon EKS de `1.21` para `1.22` e depois para `1.23`, mas não poderá atualizar o cluster para a versão `1.24` enquanto o `kubelet` estiver na versão `1.21`.
- Como prática recomendada, antes de iniciar uma atualização, verifique se o `kubelet` em seus nós está na mesma versão do Kubernetes do seu ambiente de gerenciamento.
- Se seu cluster estiver configurado com uma versão do Amazon VPC CNI plugin for Kubernetes anterior à `1.8.0`, recomendamos atualizar o plug-in para a versão mais

recente antes de atualizar seu cluster. Para atualizar o plug-in, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

- Se você estiver atualizando o cluster para uma versão `1.25` ou posterior e tiver o AWS Load Balancer Controller implantado em seu cluster, atualize o controlador para a versão `2.4.7` ou posterior antes de atualizar sua versão do cluster para `1.25`. Para obter mais informações, consulte as notas de versão do Kubernetes 1.25.

eksctl

Este procedimento exige a versão `eksctl 0.187.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar e atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Atualize a versão Kubernetes do ambiente de gerenciamento do Amazon EKS. Substitua *`my-cluster`* pelo nome do cluster. Substitua *`1.30`* pelo número da versão compatível com o Amazon EKS para a qual você deseja atualizar o cluster. Para obter a lista completa dos números das versões compatíveis, consulte Compreeender o ciclo de vida da versão do Kubernetes no EKS.

```
eksctl upgrade cluster --name my-cluster --version 1.30 --approve
```

O processo pode demorar alguns minutos para ser concluído.

AWS Management Console

a. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
b. Escolha o nome do cluster do Amazon EKS a ser atualizado e selecione Update cluster version (Atualizar versão do cluster).
c. Em KubernetesKubernetes version (Versão do Kubernetes), selecione a versão para a qual o cluster será atualizado e escolha Update (Atualizar).
d. Em Cluster name (Nome do cluster), insira o nome do seu cluster e escolha Confirm (Confirmar).

O processo pode demorar alguns minutos para ser concluído.

AWS CLI

a. Atualize o cluster do Amazon EKS com o seguinte comando da AWS CLI. Substitua *example values* pelos seus próprios valores. Substitua *1.30* pelo número da versão compatível com o Amazon EKS para a qual você deseja atualizar o cluster. Para obter a lista completa dos números das versões compatíveis, consulte Compreeender o ciclo de vida da versão do Kubernetes no EKS.

```
aws eks update-cluster-version --region region-code --name my-cluster --
kubernetes-version 1.30
```

Veja um exemplo de saída abaixo.

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.30"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
}
```

b. Monitore o status do cluster com o comando a seguir. Use o nome do cluster e o ID da atualização que o comando anterior retornou. Quando um status de `Successful` for exibido, significa que a atualização está concluída. O processo pode demorar alguns minutos para ser concluído.

```
aws eks describe-update --region region-code --name my-cluster --update-
id b5f0ba18-9a87-4450-b5a0-825e6e84496f
```

Veja um exemplo de saída abaixo.

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "Successful",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.30"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
}
```

5. Depois que a atualização do cluster for concluída, atualize os nós para a mesma versão secundária que a do Kubernetes do cluster atualizado. Para ter mais informações, consulte Atualizar nós autogerenciados para seu cluster e Atualizar um grupo de nós gerenciados para seu cluster. Todos os Pods novos iniciados no Fargate têm uma versão do `kubelet` que corresponde à versão do cluster. Os Pods existentes do Fargate não são alterados.
6. (Opcional) Se você implantou o Kubernetes Cluster Autoscaler no cluster antes de atualizá-lo, atualize o Cluster Autoscaler para a versão mais recente que corresponda à versão principal e secundária do Kubernetes para a qual você atualizou.

    a. Abra a página releases (versões) do Cluster Autoscaler em um navegador da Web e encontre a versão mais recente do Cluster Autoscaler que corresponda às versões principal e secundária do Kubernetes do cluster. Por exemplo, se a versão do Kubernetes do cluster for a versão `1.30`, localize a versão mais recente do Cluster Autoscaler que comece com

`1.30`. Registre o número de versão semântico (`1.30.n`, por exemplo) dessa versão para usar na próxima etapa.

b. Defina a tag de imagem do Cluster Autoscaler como a versão que você registrou na etapa anterior com o comando a seguir. Se necessário, substitua *`1.30.n`* pelo seu próprio valor.

```
kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-
autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:v1.30.n
```

7. (Clusters apenas com nós de GPU) Se o cluster tiver grupos de nós com suporte a GPU (por exemplo, `p3.2xlarge`), você deverá atualizar o DaemonSet do plug-in de dispositivo NVIDIA para Kubernetes no seu cluster. Substitua *`vX.X.X`* pela versão desejada de NVIDIA/k8s-device-plugin antes de executar o comando a seguir.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

8. Atualize os complementos Amazon VPC CNI plugin for Kubernetes, CoreDNS e `kube-proxy`. Recomendamos atualizar os complementos para as versões mínimas listadas nos tokens da conta de serviço.

   - Se você está usando complementos do Amazon EKS, selecione Clusters no console do Amazon EKS e, em seguida, selecione o nome do cluster que você atualizou no painel de navegação esquerdo. As notificações serão exibidas no console. Elas informam que uma nova versão está disponível para cada complemento que tenha uma atualização disponível. Para atualizar um complemento, selecione a guia Add-ons (Complementos). Em uma das caixas de um complemento que tenha uma atualização disponível, selecione Update now (Atualizar agora), selecione uma versão disponível e, em seguida, selecione Update (Atualizar).
   - Ou então, você pode usar a AWS CLI ou o `eksctl` para atualizar os complementos. Para obter mais informações, consulte Atualizar um complemento do Amazon EKS.

9. Se necessário, atualize a versão do `kubectl`. Você deve usar uma versão do `kubectl` que esteja em uma versão secundária de diferença do plano de controle do cluster do Amazon EKS. Por exemplo, um cliente do `kubectl` `1.29` funciona com clusters do Kubernetes `1.28`, `1.29` e `1.30`. É possível verificar a versão instalada atualmente com o comando a seguir.

```
kubectl version --client
```

Fazer o downgrade da versão do Kubernetes de um cluster do Amazon EKS

Não é possível fazer o downgrade do Kubernetes de um cluster do Amazon EKS. Em vez disso, crie um novo cluster em uma versão anterior do Amazon EKS e migre as workloads.

Excluir um cluster

Quando terminar de usar um cluster do Amazon EKS, você deverá excluir os recursos associados a ele para não causar custos desnecessários.

Para remover um cluster conectado, consulte Cancelar o registro de um cluster do Kubernetes no console do Amazon EKS

> **Important**
>
> - Se tiver serviços ativos no cluster associados a um load balancer, você deve excluir esses serviços antes de excluir o cluster para que os load balancers sejam excluídos corretamente. Caso contrário, você poderá ter recursos órfãos na VPC que impedirão que você exclua a VPC.
> - Se receber um erro porque o criador do cluster foi removido, consulte este artigo para resolvê-lo.
> - Os recursos do Amazon Managed Service for Prometheus estão fora do ciclo de vida do cluster e precisam ser mantidos separados do cluster. Ao excluir seu cluster, certifique-se de excluir também todos os extratores relevantes para interromper os custos aplicáveis. Para obter mais informações, consulte Encontrar e excluir extratores no Guia do usuário do Amazon Managed Service for Prometheus.

Você pode excluir um cluster com o `eksctl`, o AWS Management Console ou com a AWS CLI.

eksctl

Para excluir os nós e o cluster do Amazon EKS com o **`eksctl`**.

Este procedimento exige a versão `eksctl` `0.187.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

1. Liste todos os serviços em execução no cluster.

   ```
   kubectl get svc --all-namespaces
   ```

2. Exclua todos os serviços que têm um valor `EXTERNAL-IP` associado. Esses serviços são liderados por um balanceador de carga do Elastic Load Balancing, e você deverá excluí-los no Kubernetes para permitir que o balanceador de carga e os recursos associados sejam liberados corretamente.

   ```
   kubectl delete svc service-name
   ```

3. Exclua o cluster e os nós associados com o comando a seguir, substituindo *prod* pelo nome do cluster.

   ```
   eksctl delete cluster --name prod
   ```

   Saída:

   ```
   [#]  using region region-code
   [#]  deleting EKS cluster "prod"
   [#]  will delete stack "eksctl-prod-nodegroup-standard-nodes"
   [#]  waiting for stack "eksctl-prod-nodegroup-standard-nodes" to get deleted
   [#]  will delete stack "eksctl-prod-cluster"
   [#]  the following EKS cluster resource(s) for "prod" will be deleted: cluster.
    If in doubt, check CloudFormation console
   ```

AWS Management Console

Para excluir os nós e o cluster do Amazon EKS com o AWS Management Console.

1. Liste todos os serviços em execução no cluster.

   ```
   kubectl get svc --all-namespaces
   ```

2. Exclua todos os serviços que têm um valor `EXTERNAL-IP` associado. Esses serviços são liderados por um balanceador de carga do Elastic Load Balancing, e você deverá excluí-los no Kubernetes para permitir que o balanceador de carga e os recursos associados sejam liberados corretamente.

```
kubectl delete svc service-name
```

3. Exclua todos os grupos de nós e perfis do Fargate.

   a. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

   b. No painel de navegação à esquerda, escolha Clusters do Amazon EKS e, em seguida, na lista de clusters com guias, escolha o nome do cluster que você deseja excluir.

   c. Escolha a guia Compute (Computação) e escolha um grupo de nós para exclusão. Escolha Delete (Excluir), insira o nome do grupo de nós e depois escolha Delete (Excluir). Exclua todos os grupos de nós no cluster.

   > **Note**
   >
   > Os grupos de nós listados são somente grupos de nós gerenciados .

   d. Escolha um perfil do Fargate para excluir, selecione Delete (Excluir), insira o nome do perfil e selecione Delete (Excluir). Exclua todos os perfis do Fargate no cluster.

4. Exclua todas as pilhas de AWS CloudFormation dos nós autogerenciados.

   a. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.

   b. Escolha a pilha de nós a ser excluída e, então, escolha Delete (Excluir).

   c. Na caixa de diálogo de confirmação Delete stack (Excluir pilha), escolha Delete stack (Excluir pilha). Exclua todas as pilhas de nós autogerenciados do cluster.

5. Excluir o cluster.

   a. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

   b. escolha o cluster a ser excluído e escolha Delete (Excluir).

   c. Na tela de confirmação de excluir cluster, selecione Delete (Excluir).

6. (Opcional) Exclua a pilha do AWS CloudFormation da VPC.

a. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.

b. Selecione a pilha da VPC a ser excluída e, então, escolha Delete (Excluir).

c. Na caixa de diálogo de confirmação Delete stack (Excluir pilha), escolha Delete stack (Excluir pilha).

AWS CLI

Para excluir os nós e o cluster do Amazon EKS com o AWS CLI.

1. Liste todos os serviços em execução no cluster.

```
kubectl get svc --all-namespaces
```

2. Exclua todos os serviços que têm um valor `EXTERNAL-IP` associado. Esses serviços são liderados por um balanceador de carga do Elastic Load Balancing, e você deverá excluí-los no Kubernetes para permitir que o balanceador de carga e os recursos associados sejam liberados corretamente.

```
kubectl delete svc service-name
```

3. Exclua todos os grupos de nós e perfis do Fargate.

a. Liste os grupos de nós no cluster com o comando a seguir.

```
aws eks list-nodegroups --cluster-name my-cluster
```

> **Note**
>
> Os grupos de nós listados são somente grupos de nós gerenciados .

b. Exclua cada grupo de nós com o comando a seguir. Exclua todos os grupos de nós no cluster.

```
aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
```

c. Liste os perfis do Fargate no cluster com o comando a seguir.

```
aws eks list-fargate-profiles --cluster-name my-cluster
```

d. Exclua cada perfil do Fargate com o comando a seguir. Exclua todos os perfis do Fargate no cluster.

```
aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --
cluster-name my-cluster
```

4. Exclua todas as pilhas de AWS CloudFormation dos nós autogerenciados.

a. Liste suas pilhas do AWS CloudFormation disponíveis com o comando a seguir. Encontre o nome do modelo do nó no resultado.

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

b. Exclua cada pilha de nós com o comando a seguir, substituindo *node-stack* pelo nome da sua pilha de nós. Exclua todas as pilhas de nós autogerenciados do cluster.

```
aws cloudformation delete-stack --stack-name node-stack
```

5. Exclua o cluster com o comando a seguir, substituindo *my-cluster* pelo nome do cluster.

```
aws eks delete-cluster --name my-cluster
```

6. (Opcional) Exclua a pilha do AWS CloudFormation da VPC.

a. Liste suas pilhas do AWS CloudFormation disponíveis com o comando a seguir. Encontre o nome do modelo da VPC na saída resultante.

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

b. Exclua a pilha da VPC com o comando a seguir, substituindo *my-vpc-stack* pelo nome da pilha da VPC.

```
aws cloudformation delete-stack --stack-name my-vpc-stack
```

Controlar o acesso à rede ao endpoint do servidor de API do cluster

Este tópico ajuda a ativar o acesso privado ao endpoint do servidor de API do Kubernetes de cluster do Amazon EKS e limitar ou desativar completamente o acesso público pela Internet.

Quando você cria um cluster, o Amazon EKS cria um endpoint para o servidor gerenciado de API do Kubernetes usado para se comunicar com o cluster (usando as ferramentas de gerenciamento do Kubernetes, como `kubectl`). Por padrão, esse endpoint do servidor de API é público para a Internet, e o acesso ao servidor de API é protegido por uma combinação do AWS Identity and Access Management (IAM) e do Controle de acesso com base em função (RBAC) nativo do Kubernetes.

Você pode habilitar o acesso privado ao servidor de API do Kubernetes para que todas as comunicações entre os nós e o servidor de API fiquem na VPC. Você pode limitar os endereços IP que podem acessar o servidor de API pela Internet ou desativar completamente o acesso à Internet para o servidor de API.

> **Note**
>
> Como esse endpoint é para o servidor de API do Kubernetes, não um endpoint tradicional do AWS PrivateLink para comunicação com uma API da AWS, ele não aparece como um endpoint no console da Amazon VPC.

Ao ativar o acesso privado ao endpoint para o cluster, o Amazon EKS cria uma zona hospedada privada do Route 53 em seu nome e a associa à VPC do cluster. Essa zona hospedada privada é gerenciada pelo Amazon EKS e não aparece nos recursos do Route 53 da sua conta. Para que a zona hospedada privada roteie o tráfego adequadamente para o seu servidor da API, a VPC deve ter `enableDnsHostnames` e `enableDnsSupport` definidos como `true`, e o conjunto de opções DHCP para sua VPC deve incluir `AmazonProvidedDNS` na lista de servidores de nome de domínio. Para obter mais informações, consulte Updating DNS support for your VPC (Atualizar o suporte de DNS para VPC) no Manual do usuário da Amazon VPC.

É possível definir os requisitos de acesso ao endpoint do servidor de API ao criar um cluster, e você pode atualizar o acesso ao endpoint do servidor de API para um cluster a qualquer momento.

Modificar o acesso ao endpoint do cluster

Use os procedimentos desta seção para modificar o acesso ao endpoint para um cluster existente. A tabela a seguir mostra as combinações compatíveis de acesso ao endpoint do servidor de API e seus comportamentos associados.

Opções de acesso ao endpoint do servidor de API

Acesso público ao endpoint	Acesso privado ao endpoint	Comportamento
Habilitado	Desabilitado	• Esse é o comportamento padrão para novos clusters do Amazon EKS. • As solicitações de API do Kubernetes que são originadas na VPC do cluster (como um nó para a comunicação do ambiente de gerenciamento) deixam a VPC, mas não a rede da Amazon. • O servidor de API do cluster é acessível pela internet. Também é possível limitar os blocos CIDR que podem acessar o endpoint público. Se você limitar o acesso a blocos CIDR específicos, é recomendável habilitar também o endpoint privado ou garantir que os blocos CIDR especificados incluam os endereços dos quais os nós e os Pods do Fargate (se você usá-los) acessam o endpoint público.

Acesso público ao endpoint	Acesso privado ao endpoint	Comportamento
Habilitado	Habilitado	• As solicitações de API do Kubernetes na VPC do cluster (como um nó para comunicação do ambiente de gerenciamento) usam o endpoint da VPC privado. • O servidor de API do cluster é acessível pela internet. Também é possível limitar os blocos CIDR que podem acessar o endpoint público.

Acesso público ao endpoint	Acesso privado ao endpoint	Comportamento
Desabilitado	Habilitado	• Todo o tráfego para o servidor de API do cluster deve vir da VPC do cluster ou de uma rede conectada. • Não há acesso público ao servidor de API pela internet. Todos os comandos `kubectl` devem vir da VPC ou de uma rede conectada. Para ver as opções de conectividade, consulte Acessar um servidor de API somente privado. • O endpoint do servidor de API do cluster é resolvido por servidores DNS públicos para um endereço IP privado da VPC. No passado, o endpoint só podia ser resolvido a partir da VPC. Se o endpoint não for resolvido para um endereço IP privado na VPC para um cluster existente, será possível: • Ativar o acesso público e desativá-lo novamente . Só é necessário fazer isso uma vez para um cluster e o endpoint será resolvido para um

Acesso público ao endpoint	Acesso privado ao endpoint	Comportamento
		endereço IP privado desse ponto em diante. • Atualizar o cluster.

É possível modificar o acesso ao endpoint do servidor de API de cluster usando o AWS Management Console ou a AWS CLI.

AWS Management Console

Para modificar o acesso ao endpoint do servidor de API do cluster usando a AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do cluster para exibir as informações dele.
3. Escolha a guia Networking (Redes) e selecione Update (Atualizar).
4. Em Private access (Acesso privado): escolha se deseja habilitar ou desabilitar o acesso privado para o endpoint do servidor de API do Kubernetes do cluster. Se você habilitar o acesso privado, as solicitações de API do Kubernetes originadas na VPC do cluster usarão o endpoint da VPC privado. É necessário habilitar o acesso privado para desabilitar o acesso público.
5. Em Public access (Acesso público): escolha se deseja habilitar ou desabilitar o acesso público para o endpoint do servidor de API do Kubernetes do cluster. Se você desabilitar o acesso público, o servidor de API do Kubernetes do cluster só poderá receber solicitações da VPC do cluster.
6. (Opcional) Se você habilitou o Public access (Acesso público), poderá especificar quais endereços da Internet podem se comunicar com o endpoint público. Selecione Advanced Settings (Configurações avançadas). Insira um bloco CIDR, como `203.0.113.5/32`. O bloco não pode incluir endereços reservados. É possível inserir blocos adicionais selecionando Add Source (Adicionar origem). Há um número máximo de blocos CIDR que você pode especificar. Para ter mais informações, consulte Visualizar e gerenciar o Amazon EKS e as cotas de serviço do Fargate. Se você não especificar nenhum bloco, o endpoint do servidor de API público receberá solicitações de todos os endereços IP (`0.0.0.0/0`). Se você restringir o acesso ao endpoint público usando blocos CIDR, é recomendável habilitar também o acesso ao endpoint privado para que os nós e os Pods do Fargate (se você usá-

los) possam se comunicar com o cluster. Sem o endpoint privado habilitado, suas origens CIDR de endpoint de acesso público devem incluir as origens de saída de sua VPC. Por exemplo, se você tiver um nó em uma sub-rede privada que se comunica com a Internet por meio de um gateway NAT, será necessário adicionar o endereço IP de saída do gateway NAT como parte de um bloco CIDR na lista de permissões no endpoint público.

7. Selecione Update (Atualizar) para concluir.

AWS CLI

Para modificar o acesso ao endpoint do servidor de API do cluster usando a AWS CLI

Conclua as etapas a seguir usando a AWS CLI versão `1.27.160` ou posterior. É possível verificar sua versão atual com `aws --version`. Para instalar ou atualizar a AWS CLI, consulte Instalar a AWS CLI.

1. Atualize o acesso ao endpoint do servidor de API do cluster com o comando da AWS CLI a seguir. Substitua o nome do cluster e os valores desejados de acesso ao endpoint. Se você definir `endpointPublicAccess=true`, também poderá inserir um único bloco CIDR ou uma lista de blocos CIDR separados por vírgulas para `publicAccessCidrs`. Os blocos não podem incluir endereços reservados. Se você especificar blocos CIDR, o endpoint do servidor de API público só receberá solicitações dos blocos listados. Há um número máximo de blocos CIDR que você pode especificar. Para ter mais informações, consulte Visualizar e gerenciar o Amazon EKS e as cotas de serviço do Fargate. Se você restringir o acesso ao endpoint público usando blocos CIDR, é recomendável habilitar também o acesso ao endpoint privado para que os nós e os Pods do Fargate (se você usá-los) possam se comunicar com o cluster. Sem o endpoint privado habilitado, suas origens CIDR de endpoint de acesso público devem incluir as origens de saída de sua VPC. Por exemplo, se você tiver um nó em uma sub-rede privada que se comunica com a Internet por meio de um gateway NAT, será necessário adicionar o endereço IP de saída do gateway NAT como parte de um bloco CIDR na lista de permissões no endpoint público. Se você não especificar nenhum bloco CIDR, o endpoint do servidor de API público receberá solicitações de todos os endereços IP (0.0.0.0/0).

> **Note**
>
> O comando a seguir permite acesso privado e acesso público a partir de um único endereço IP para o endpoint do servidor de API. Substitua *`203.0.113.5/32`* por

um único bloco CIDR ou uma lista de blocos CIDR separados por vírgulas aos quais você deseja restringir o acesso à rede.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --resources-vpc-config
 endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

Veja um exemplo de saída abaixo.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

2. Monitore o status da atualização do acesso ao endpoint com o comando a seguir, usando o nome do cluster e o ID da atualização retornado pelo comando anterior. Sua atualização estará concluída quando o status for exibido como `Successful`.

```
aws eks describe-update \
```

```
    --region region-code \
    --name my-cluster \
    --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000
```

Veja um exemplo de saída abaixo.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "Successful",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

Acessar um servidor de API somente privado

Se tiver desabilitado o acesso público ao endpoint do servidor de API do Kubernetes do cluster, você só poderá acessar o servidor de API pela VPC ou por uma rede conectada. Veja a seguir algumas maneiras possíveis de acessar o endpoint do servidor de API do Kubernetes:

Rede conectada

Conecte a sua rede à VPC com um gateway de trânsito da AWS ou outra opção de conectividade e depois use um computador na rede conectada. É necessário garantir que o grupo de segurança

do plano de controle do Amazon EKS tenha regras para permitir o tráfego de entrada na porta 443 da rede conectada.

Bastion host do Amazon EC2

Você pode executar uma instância do Amazon EC2 em uma sub-rede pública na VPC do cluster e fazer login via SSH nessa instância para executar os comandos do `kubectl`. Para obter mais informações, consulte Bastion hosts do Linux na AWS. É necessário garantir que o grupo de segurança do plano de controle do Amazon EKS tenha regras para permitir o tráfego de entrada na porta 443 do seu bastion host. Para ter mais informações, consulte Considerações e requisitos sobre grupos de segurança do Amazon EKS.

Quando você configurar `kubectl` para o bastion host, use credenciais da AWS que já estiverem mapeadas para a configuração do RBAC do cluster ou adicione a entidade principal do IAM que o bastion usará para a configuração do RBAC antes de remover o acesso público ao endpoint. Para ter mais informações, consulte the section called “Conceder acesso a APIs do Kubernetes” e Acesso negado ou não autorizado (`kubectl`).

IDE AWS Cloud9

AWS Cloud9 é um ambiente de desenvolvimento integrado (IDE) baseado na nuvem que permite escrever, executar e depurar código com apenas um navegador. Você pode criar um IDE do AWS Cloud9 na VPC do cluster e usar o IDE para se comunicar com o cluster. Para obter mais informações, consulte Criar um ambiente no AWS Cloud9. É necessário garantir que o grupo de segurança do plano de controle do Amazon EKS contenha regras para permitir o tráfego de entrada na porta 443 do seu grupo de segurança IDE. Para ter mais informações, consulte Considerações e requisitos sobre grupos de segurança do Amazon EKS.

Ao configurar `kubectl` para o IDE do AWS Cloud9, use credenciais da AWS que já estiverem mapeadas para a configuração do RBAC do cluster ou adicione a entidade principal do IAM que o IDE usará para a configuração do RBAC antes de remover o acesso público ao endpoint. Para ter mais informações, consulte Conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes e Acesso negado ou não autorizado (`kubectl`).

Criptografar segredos do Kubernetes com o AWS KMS em clusters existentes

Se você habilitar a criptografia de segredos, os segredos do Kubernetes serão criptografados usando a AWS KMS key selecionada. A chave do KMS deve atender às seguintes condições:

- Simétrica
- Pode criptografar e descriptografar dados
- Criação concluída na mesma Região da AWS do cluster
- Se a chave do KMS tiver sido criada em uma conta diferente, a entidade principal do IAM deverá ter acesso à chave do KMS.

Para obter mais informações, consulte Permitir que entidades principais do IAM em outras contas usem uma chave do KMS no Guia do desenvolvedor do AWS Key Management Service.

> **Warning**
>
> Não é possível desativar a criptografia de segredos depois de habilitá-la. Essa ação é irreversível.

eksctl

É possível ativar a criptografia de duas formas:

- Adicione criptografia ao cluster com um único comando.

  Para recriptografar seus segredos automaticamente, execute o comando a seguir.

  ```
  eksctl utils enable-secrets-encryption \
      --cluster my-cluster \
      --key-arn arn:aws:kms:region-code:account:key/key
  ```

  Para optar por não recriptografar seus segredos automaticamente, execute o comando a seguir.

  ```
  eksctl utils enable-secrets-encryption
      --cluster my-cluster \
      --key-arn arn:aws:kms:region-code:account:key/key \
      --encrypt-existing-secrets=false
  ```

- Adicione criptografia ao cluster com um arquivo `kms-cluster.yaml`.

  ```
  apiVersion: eksctl.io/v1alpha5
  kind: ClusterConfig
  ```

```
metadata:
  name: my-cluster
  region: region-code

secretsEncryption:
  keyARN: arn:aws:kms:region-code:account:key/key
```

Para que seus segredos sejam recriptografados automaticamente, execute o comando a seguir.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

Para optar por não recriptografar seus segredos automaticamente, execute o comando a seguir.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-
secrets=false
```

AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha o cluster ao qual você deseja adicionar a criptografia do KMS.
3. Escolha a guia Overview (Visão geral) (selecionada por padrão).
4. Role para baixo até Secrets encryption (Criptografia de segredos) e escolha Enable (Habilitar).
5. Selecione uma chave na lista suspenso e escolha Enable (Habilitar). Se nenhuma chave estiver listada, primeiro você deve criar uma. Para obter mais informações, consulte Criar chaves
6. Escolha o botão Confirm (Confirmar) para usar a chave escolhida.

AWS CLI

1. Associe a configuração da criptografia de segredos ao seu cluster usando o seguinte comando da AWS CLI: Substitua *example values* pelos seus próprios valores.

```
aws eks associate-encryption-config \
```

```
    --cluster-name my-cluster \
    --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws:kms:region-code:account:key/key"}}]'
```

Veja um exemplo de saída abaixo.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734,
    "errors": []
  }
}
```

2. Você pode monitorar o status da atualização da criptografia com o comando a seguir. Utilize o `cluster name` e o `update ID` específicos que foram retornados na saída anterior. Quando um status de `Successful` for exibido, significa que a atualização está concluída.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id 3141b835-8103-423a-8e68-12c2521ffa4d
```

Veja um exemplo de saída abaixo.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
```

```
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734>,
    "errors": []
  }
}
```

3. Para verificar se a criptografia está habilitada no cluster, execute comando `describe-cluster`. A resposta contém uma string `EncryptionConfig`.

```
aws eks describe-cluster --region region-code --name my-cluster
```

Depois de habilitar a criptografia no cluster, você precisa criptografar todos os segredos existentes com a nova chave:

> **Note**
>
> Se você utilizar `eksctl`, apenas será necessário executar o seguinte comando se você optar por não recriptografar seus segredos automaticamente:

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-
encryption-timestamp="time value"
```

> **Warning**
>
> Se você habilitar a criptografia de segredos para um cluster existente e se algum dia a chave do KMS usada for excluída, não haverá maneira de recuperar o cluster. Se você excluir a chave do KMS, colocará permanentemente o cluster em um estado degradado. Para informações, consulte Exclusão das chaves do AWS KMS.

> **Note**
>
> Por padrão, o comando `create-key` cria uma chave do KMS simétrica de criptografia com uma política de chave que dá acesso ao administrador raiz da conta às ações e recursos do

AWS KMS. Se você quiser reduzir o escopo das permissões, certifique-se de que as ações `kms:DescribeKey` e `kms:CreateGrant` sejam permitidas na política para a entidade principal que chama a API `create-cluster`.
Para clusters que usam criptografia envelopada do KMS, são necessárias permissões `kms:CreateGrant`. A condição `kms:GrantIsForAWSResource` não tem suporte para a ação CreateCluster e não deve ser utilizada nas políticas do KMS para controlar permissões `kms:CreateGrant` de usuários que executam CreateCluster.

Implantar nós do Windows em clusters do EKS

Antes de implantar nós do Windows, esteja ciente das considerações a seguir.

Considerações

- É possível usar rede de hosts nos nós do Windows usando `HostProcess` Pods. Para obter mais informações, consulte Criar um Windows `HostProcessPod` na documentação do Kubernetes.
- Os clusters do Amazon EKS devem conter um ou mais nós do Linux ou do Fargate para executar Pods do sistema core que só são executados no Linux, como o CoreDNS.
- Os logs de eventos do `kubelet` e `kube-proxy` são redirecionados para o log de eventos do EKS Windows e são definidos com um limite de 200 MB.
- Não é possível usar Grupos de segurança do Pods com Pods executados em nós do Windows.
- Não é possível usar redes personalizadas com nós do Windows.
- Não é possível usar `IPv6` com nós do Windows.
- Os nós do Windows são compatíveis com uma interface de rede elástica por nó. Por padrão, o número de Pods que podem ser executados por nó do Windows é igual ao número de endereços IP disponíveis por interface de rede elástica para o tipo de instância do nó, menos um. Para obter mais informações, consulte Endereços IP por interface de rede por tipo de instância no Guia do usuário do Amazon EC2.
- Em um cluster do Amazon EKS, um único serviço com um balanceador de carga é compatível com até 1024 Pods de backend. Cada Pod tem seu próprio endereço IP exclusivo. O limite anterior de 64 Pods não é mais o caso, depois de uma atualização do Windows Server a partir da Compilação do sistema operacional 17763.2746.
- Contêineres do Windows não são compatíveis com Pods do Amazon EKS no Fargate.

- Não é possível recuperar logs do pod `vpc-resource-controller`. Antes, era possível implantar o controlador no plano de dados.
- Há um período de resfriamento antes de um endereço `IPv4` ser atribuído a um novo pod. Isso evita que o tráfego flua para um pod antigo com o mesmo endereço `IPv4` por causa de regras `kube-proxy` obsoletas.
- A fonte do controlador é gerenciada no GitHub. Para postar sua contribuição ou arquivar problemas sobre o controlador, acesse o projeto no GitHub.
- Ao especificar uma ID de AMI personalizada para grupos de nós gerenciados do Windows, adicione `eks:kube-proxy-windows` ao mapa de configuração do AWS IAM Authenticator. Para obter mais informações, consulte Limites e condições ao especificar uma ID de AMI.
- Se preservar os endereços IPv4 disponíveis for crucial para sua sub-rede, consulte IP Address Management em Windows Networking no Guia de práticas recomendadas do EKS para obter orientação.

Pré-requisitos

- Um cluster existente. O cluster deve estar executando uma das versões do Kubernetes e versões da plataforma listadas na tabela a seguir. É compatível também com todas as versões do Kubernetes e da plataforma posteriores às versões listadas. Se a versão do cluster ou da plataforma for anterior a uma das versões a seguir, será necessário habilitar o suporte ao Windows herdado no plano de dados do cluster. Depois que o cluster estiver em uma das versões a seguir do Kubernetes e da plataforma, ou versões posteriores, será possível remover o suporte ao Windowsherdado e habilitar o suporte ao Windows no ambiente de gerenciamento.

Versão do Kubernetes	Versão da plataforma
1.30	eks.2
1.29	eks.1
1.28	eks.1
1.27	eks.1
1.26	eks.1

Versão do Kubernetes	Versão da plataforma
1.25	eks.1
1.24	eks.2

- O cluster deve ter pelo menos um (recomendamos pelo menos dois) nó do Linux ou Pod do Fargate para executar o CoreDNS. Se você habilitar o suporte ao Windows legado, deverá usar um nó do Linux (não é possível usar um Pod do Fargate) para executar o CoreDNS.
- Um Função do IAM do cluster do Amazon EKS existente.

Habilitar o suporte do Windows

Se o cluster não estiver em uma das versões do Kubernetes e da plataforma listadas em Pré-requisitos, ou versões posteriores, será necessário habilitar o suporte ao Windows herdado. Para obter mais informações, consulte Suporte a clusters herdados para nós Windows.

Se você nunca habilitou o suporte ao Windows no cluster, vá para a próxima etapa.

Se você habilitou o suporte ao Windows em um cluster de uma versão anterior à versão do Kubernetes ou da plataforma listada em Prerequisites (Pré-requisitos), primeiro será necessário remover `vpc-resource-controller` e `vpc-admission-webhook` do plano de dados. Eles estão defasados e não são mais necessários.

Para habilitar o suporte ao Windows no cluster

1. Se não houver nós do Amazon Linux no cluster e você usar grupos de segurança para Pods, pule para a próxima etapa. Caso contrário, confirme se a política gerenciada `AmazonEKSVPCResourceController` está anexada à função do cluster. Substitua o *`eksClusterRole`* pelo nome do cluster.

   ```
   aws iam list-attached-role-policies --role-name eksClusterRole
   ```

   Veja um exemplo de saída abaixo.

   ```
   {
       "AttachedPolicies": [
           {
               "PolicyName": "AmazonEKSClusterPolicy",
   ```

```
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
        },
        {
            "PolicyName": "AmazonEKSVPCResourceController",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"
        }
    ]
}
```

Se a política estiver anexada, como está na saída anterior, pule a próxima etapa.

2. Anexe a política gerenciada AmazonEKSVPCResourceController a Função do IAM do cluster do Amazon EKS. Substitua o *eksClusterRole* pelo nome do cluster.

```
aws iam attach-role-policy \
  --role-name eksClusterRole \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
```

3. Crie um arquivo denominado *vpc-resource-controller-configmap.yaml* com o seguinte conteúdo:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-windows-ipam: "true"
```

4. Aplicar o `ConfigMap` ao seu cluster.

```
kubectl apply -f vpc-resource-controller-configmap.yaml
```

5. Verifique se seu `ConfigMap` do `aws-auth` contém um mapeamento para o perfil de instância do nó Windows para incluir o grupo de permissões do RBAC `eks:kube-proxy-windows`. O comando a seguir pode ser executado para verificar.

```
kubectl get configmap aws-auth -n kube-system -o yaml
```

Veja um exemplo de saída abaixo.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      - eks:kube-proxy-windows # This group is required for Windows DNS resolution
 to work
      rolearn: arn:aws:iam::111122223333:role/eksNodeRole
      username: system:node:{{EC2PrivateDNSName}}
[...]
```

Você deverá ver `eks:kube-proxy-windows` listado em grupos. Se o grupo não for especificado, será necessário atualizar seu `ConfigMap` ou criá-lo para incluir o grupo necessário. Para obter mais informações sobre o `ConfigMap` do `aws-auth`, consulte Como aplicar o `ConfigMap``aws-auth` ao seu cluster.

Implantar pods do Windows

Para implantar pods no cluster, será necessário especificar o sistema operacional que eles usam, se você estiver executando uma mistura de tipos de nós.

Para Pods do Linux, use o texto do seletor de nó a seguir nos manifestos.

```
nodeSelector:
        kubernetes.io/os: linux
        kubernetes.io/arch: amd64
```

Para Pods do Windows, use o texto do seletor de nó a seguir nos manifestos.

```
nodeSelector:
        kubernetes.io/os: windows
        kubernetes.io/arch: amd64
```

Você pode implantar uma aplicação de exemplo para ver os seletores de nós em uso.

Oferecer suporte a uma maior densidade de Pod em nós do Windows

No Amazon EKS, cada Pod recebe um endereço `IPv4` da sua VPC. Como resultado, o número de Pods que podem ser implantados em um nó é limitado pelos endereços IP disponíveis, mesmo que haja recursos suficientes para executar mais Pods no nó. Como somente uma interface de rede elástica é aceita por cada nó do Windows, por padrão, o número máximo de endereços IP disponíveis em um nó do Windows é igual a:

```
Number of private IPv4 addresses for each interface on the node - 1
```

Um endereço IP é usado como endereço IP principal da interface de rede, e portanto, não pode ser alocado a Pods.

É possível permitir uma maior densidade de Pod em nós do Windows com a habilitação da delegação de prefixo IP. Esse recurso permite atribuir um prefixo `IPv4 /28` à interface de rede primária, em vez de atribuir endereços `IPv4` secundários. A atribuição de um prefixo IP aumenta o número máximo de endereços `IPv4` disponíveis no nó para:

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) *
 16
```

Com esse número significativamente maior de endereços IP disponíveis, os endereços IP disponíveis não devem limitar sua capacidade de escalar o número de Pods em seus nós. Para obter mais informações, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2.

Desabilitar o suporte a Windows

Para desabilitar o suporte ao Windows no cluster

1. Se o cluster contiver nós do Amazon Linux e você usar grupos de segurança para Pods com eles, pule esta etapa.

   Remova a política do IAM `AmazonVPCResourceController` gerenciada da função do cluster. Substitua *`eksClusterRole`* pelo nome da função do cluster e *`111122223333`* pelo ID da conta.

   ```
   aws iam detach-role-policy \
   ```

```
    --role-name eksClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
```

2. Desabilitar Windows IPAM no ConfigMap do `amazon-vpc-cni`.

```
kubectl patch configmap/amazon-vpc-cni \
                    -n kube-system \
                    --type merge \
                    -p '{"data":{"enable-windows-ipam":"false"}}'
```

Suporte a clusters herdados para nós Windows

Antes de implantar nós do Windows, esteja ciente das considerações a seguir.

Considerações

- É possível usar rede de hosts nos nós do Windows usando `HostProcess` Pods. Para obter mais informações, consulte Criar um Windows `HostProcess`Pod na documentação do Kubernetes.
- Os clusters do Amazon EKS devem conter um ou mais nós do Linux ou do Fargate para executar Pods do sistema core que só são executados no Linux, como o CoreDNS.
- Os logs de eventos do `kubelet` e `kube-proxy` são redirecionados para o log de eventos do `EKS` Windows e são definidos com um limite de 200 MB.
- Não é possível usar Grupos de segurança do Pods com Pods executados em nós do Windows.
- Não é possível usar redes personalizadas com nós do Windows.
- Não é possível usar `IPv6` com nós do Windows.
- Os nós do Windows são compatíveis com uma interface de rede elástica por nó. Por padrão, o número de Pods que podem ser executados por nó do Windows é igual ao número de endereços IP disponíveis por interface de rede elástica para o tipo de instância do nó, menos um. Para obter mais informações, consulte Endereços IP por interface de rede por tipo de instância no Guia do usuário do Amazon EC2.
- Em um cluster do Amazon EKS, um único serviço com um balanceador de carga é compatível com até 1024 Pods de backend. Cada Pod tem seu próprio endereço IP exclusivo. O limite anterior de 64 Pods não é mais o caso, depois de uma atualização do Windows Server a partir da Compilação do sistema operacional 17763.2746.
- Contêineres do Windows não são compatíveis com Pods do Amazon EKS no Fargate.

- Não é possível recuperar logs do pod `vpc-resource-controller`. Antes, era possível implantar o controlador no plano de dados.
- Há um período de resfriamento antes de um endereço `IPv4` ser atribuído a um novo pod. Isso evita que o tráfego flua para um pod antigo com o mesmo endereço `IPv4` por causa de regras `kube-proxy` obsoletas.
- A fonte do controlador é gerenciada no GitHub. Para postar sua contribuição ou arquivar problemas sobre o controlador, acesse o projeto no GitHub.
- Ao especificar uma ID de AMI personalizada para grupos de nós gerenciados do Windows, adicione `eks:kube-proxy-windows` ao mapa de configuração do AWS IAM Authenticator. Para obter mais informações, consulte Limites e condições ao especificar uma ID de AMI.
- Se preservar os endereços IPv4 disponíveis for crucial para sua sub-rede, consulte IP Address Management em Windows Networking no Guia de práticas recomendadas do EKS para obter orientação.

Pré-requisitos

- Um cluster existente. O cluster deve estar executando uma das versões do Kubernetes e versões da plataforma listadas na tabela a seguir. É compatível também com todas as versões do Kubernetes e da plataforma posteriores às versões listadas. Se a versão do cluster ou da plataforma for anterior a uma das versões a seguir, será necessário habilitar o suporte ao Windows herdado no plano de dados do cluster. Depois que o cluster estiver em uma das versões a seguir do Kubernetes e da plataforma, ou versões posteriores, será possível remover o suporte ao Windowsherdado e habilitar o suporte ao Windows no ambiente de gerenciamento.

Versão do Kubernetes	Versão da plataforma
1.30	eks.2
1.29	eks.1
1.28	eks.1
1.27	eks.1
1.26	eks.1
1.25	eks.1

Versão do Kubernetes	Versão da plataforma
1.24	eks.2

- O cluster deve ter pelo menos um (recomendamos pelo menos dois) nó do Linux ou Pod do Fargate para executar o CoreDNS. Se você habilitar o suporte ao Windows legado, deverá usar um nó do Linux (não é possível usar um Pod do Fargate) para executar o CoreDNS.
- Um Função do IAM do cluster do Amazon EKS existente.

Remover o suporte ao Windows herdado do plano de dados

Se você habilitou o suporte ao Windows em um cluster de uma versão anterior à versão do Kubernetes ou da plataforma listada em Prerequisites (Pré-requisitos), primeiro será necessário remover `vpc-resource-controller` e `vpc-admission-webhook` do plano de dados. Eles estão defasados e não são mais necessários, pois a funcionalidade que forneciam agora está habilitada no ambiente de gerenciamento.

1. Desinstale `vpc-resource-controller` usando o comando a seguir. Use esse comando independentemente da ferramenta com a qual tenha sido instalado originalmente. Substitua *`region-code`* (somente a instância desse texto depois de `/manifests/`) pela Região da AWS na qual está o cluster.

   ```
   kubectl delete -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-controller.yaml
   ```

2. Desinstale o `vpc-admission-webhook` usando as instruções para a ferramenta com a qual você instalou.

   eksctl

   Execute os seguintes comandos.

   ```
   kubectl delete deployment -n kube-system vpc-admission-webhook
   kubectl delete service -n kube-system vpc-admission-webhook
   kubectl delete mutatingwebhookconfigurations.admissionregistration.k8s.io vpc-admission-webhook-cfg
   ```

kubectl on macOS or Windows

Execute o seguinte comando . Substitua *region-code* (somente a instância desse texto depois de `/manifests/`) pela Região da AWS na qual está o cluster.

```
kubectl delete -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-deployment.yaml
```

3. Habilite o suporte ao Windows para o cluster no ambiente de gerenciamento.

Habilitar suporte a Windows herdado

Se o cluster estiver em uma das versões do Kubernetes e da plataforma listadas em Prerequisites (Pré-requisitos), ou em versões posteriores, recomendamos habilitar o suporte ao Windows no ambiente de gerenciamento. Para obter mais informações, consulte Habilitar o suporte do Windows.

As etapas a seguir ajudam a habilitar o suporte ao Windows herdado para o plano de dados do cluster do Amazon EKS, caso a versão do cluster ou da plataforma seja anterior às versões listadas em Prerequisites (Pré-requisitos). Depois que as versões do cluster e da plataforma forem uma das versões listadas em Prerequisites (Pré-requisitos), ou versões posteriores, recomendamos remover o suporte ao Windows herdado e habilitá-lo para o ambiente de gerenciamento.

Você pode usar `eksctl`, um cliente do Windows ou um cliente do macOS ou Linux para habilitar o suporte ao Windows herdado para o cluster.

eksctl

Para habilitar o suporte ao Windows herdado para o cluster com **`eksctl`**

Pré-requisito

Este procedimento exige a versão `eksctl` `0.187.0` ou superior. Você pode verificar a versão com o comando a seguir.

```
eksctl version
```

Para obter mais informações sobre a instalação ou atualização do `eksctl` consulte Instalação na documentação do `eksctl`.

1. Habilite o suporte ao Windows para o cluster do Amazon EKS com o comando do `eksctl` a seguir. Substitua o *`my-cluster`* pelo nome do cluster. Esse comando implanta o controlador de recursos da VPC e o webhook do controlador de admissão da VPC que são necessários nos clusters do Amazon EKS para executar as workloads do Windows.

```
eksctl utils install-vpc-controllers --cluster my-cluster --approve
```

> **Important**
>
> O webhook do controlador de admissão da VPC é assinado com um certificado que expira um ano após a data de emissão. Para evitar tempo de inatividade, certifique-se de renovar o certificado antes que ele expire. Para obter mais informações, consulte Renovar o certificado webhook de admissão da VPC.

2. Depois de habilitar o suporte ao Windows, você pode iniciar um grupo de nós do Windows no cluster. Para obter mais informações, consulte Criar nós Microsoft Windows autogerenciados.

Windows

Para habilitar o suporte ao Windows herdado para o cluster com um cliente do Windows

Nas etapas a seguir, substitua *`region-code`* pela Região da AWS na qual o cluster reside.

1. Implante o controlador de recursos da VPC em seu cluster.

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-controller.yaml
```

2. Implante o webhook do controlador de admissão da VPC em seu cluster.

   a. Faça download dos scripts e dos arquivos de implantação necessários.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-deployment.yaml;
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/Setup-VPCAdmissionWebhook.ps1;
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.ps1;
```

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/
vpc-admission-webhook/latest/webhook-patch-ca-bundle.ps1;
```

b. Instale o OpenSSL e o jq.

c. Configure e implante o webhook de admissão da VPC.

```
./Setup-VPCAdmissionWebhook.ps1 -DeploymentTemplate ".\vpc-admission-
webhook-deployment.yaml"
```

> **Important**
>
> O webhook do controlador de admissão da VPC é assinado com um certificado que expira um ano após a data de emissão. Para evitar tempo de inatividade, certifique-se de renovar o certificado antes que ele expire. Para obter mais informações, consulte Renovar o certificado webhook de admissão da VPC.

3. Determine se o cluster tem a vinculação de função de cluster necessária.

```
kubectl get clusterrolebinding eks:kube-proxy-windows
```

Se uma saída semelhante ao resultado de exemplo a seguir for retornada, o cluster terá a vinculação de função necessária.

```
NAME                       AGE
eks:kube-proxy-windows     10d
```

Se a saída incluir `Error from server (NotFound)`, o cluster não tem a associação de função de cluster necessária. Adicione a vinculação criando um arquivo chamado *eks-kube-proxy-windows-crb.yaml* com o conteúdo a seguir.

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: eks:kube-proxy-windows
  labels:
    k8s-app: kube-proxy
    eks.amazonaws.com/component: kube-proxy
subjects:
  - kind: Group
```

```
    name: "eks:kube-proxy-windows"
roleRef:
  kind: ClusterRole
  name: system:node-proxier
  apiGroup: rbac.authorization.k8s.io
```

Aplique a configuração ao cluster.

```
kubectl apply -f eks-kube-proxy-windows-crb.yaml
```

4. Depois de habilitar o suporte ao Windows, você pode iniciar um grupo de nós do Windows no cluster. Para obter mais informações, consulte Criar nós Microsoft Windows autogerenciados.

macOS and Linux

Para habilitar o suporte ao Windows herdado para seu cluster com um cliente do macOS ou Linux

Este procedimento requer que a biblioteca `openssl` e o processador `jq` JSON estejam instalados em seu sistema cliente.

Nas etapas a seguir, substitua *`region-code`* pela Região da AWS na qual o cluster reside.

1. Implante o controlador de recursos da VPC em seu cluster.

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-controller.yaml
```

2. Crie o manifesto webhook do controlador de admissão da VPC para seu cluster.

   a. Faça download dos scripts e dos arquivos de implantação necessários.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.sh
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-patch-ca-bundle.sh
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-deployment.yaml
```

   b. Adicione permissões aos scripts de shell para que eles possam ser executados.

```
chmod +x webhook-create-signed-cert.sh webhook-patch-ca-bundle.sh
```

c. Crie um segredo para comunicação segura.

```
./webhook-create-signed-cert.sh
```

d. Verifique o segredo.

```
kubectl get secret -n kube-system vpc-admission-webhook-certs
```

e. Configure o webhook e crie um arquivo de implantação.

```
cat ./vpc-admission-webhook-deployment.yaml | ./webhook-patch-ca-bundle.sh >
 vpc-admission-webhook.yaml
```

3. Implante o webhook de admissão da VPC.

```
kubectl apply -f vpc-admission-webhook.yaml
```

> **Important**
>
> O webhook do controlador de admissão da VPC é assinado com um certificado que expira um ano após a data de emissão. Para evitar tempo de inatividade, certifique-se de renovar o certificado antes que ele expire. Para obter mais informações, consulte Renovar o certificado webhook de admissão da VPC.

4. Determine se o cluster tem a vinculação de função de cluster necessária.

```
kubectl get clusterrolebinding eks:kube-proxy-windows
```

Se uma saída semelhante ao resultado de exemplo a seguir for retornada, o cluster terá a vinculação de função necessária.

```
NAME                     ROLE                                AGE
eks:kube-proxy-windows   ClusterRole/system:node-proxier   19h
```

Se a saída incluir `Error from server (NotFound)`, o cluster não tem a associação de função de cluster necessária. Adicione a vinculação criando um arquivo chamado *`eks-kube-proxy-windows-crb.yaml`* com o conteúdo a seguir.

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: eks:kube-proxy-windows
  labels:
    k8s-app: kube-proxy
    eks.amazonaws.com/component: kube-proxy
subjects:
  - kind: Group
    name: "eks:kube-proxy-windows"
roleRef:
  kind: ClusterRole
  name: system:node-proxier
  apiGroup: rbac.authorization.k8s.io
```

Aplique a configuração ao cluster.

```
kubectl apply -f eks-kube-proxy-windows-crb.yaml
```

5. Depois de habilitar o suporte ao Windows, você pode iniciar um grupo de nós do Windows no cluster. Para obter mais informações, consulte Criar nós Microsoft Windows autogerenciados.

Renovar o certificado webhook de admissão da VPC

O certificado usado pelo webhook de admissão da VPC expira um ano após a emissão. Para evitar o tempo de inatividade, é importante renovar o certificado antes que ele expire. Você pode verificar a data de validade do certificado atual com o comando a seguir.

```
kubectl get secret \
    -n kube-system \
    vpc-admission-webhook-certs -o json | \
    jq -r '.data."cert.pem"' | \
    base64 -decode | \
    openssl x509 \
    -noout \
    -enddate | \
    cut -d= -f2
```

Veja um exemplo de saída abaixo.

```
May 28 14:23:00 2022 GMT
```

Você pode renovar o certificado usando `eksctl` ou um computador com Windows ou Linux/macOS. Siga as instruções para a ferramenta que você usou originalmente para instalar o webhook de admissão da VPC. Por exemplo, se você instalou originalmente o webhook de admissão da VPC usando `eksctl`, você deverá renovar o certificado usando as instruções na guia `eksctl`.

eksctl

1. Instale o certificado Substitua o *`my-cluster`* pelo nome do cluster.

   ```
   eksctl utils install-vpc-controllers -cluster my-cluster -approve
   ```

2. Verifique se você recebe o seguinte resultado:

   ```
   2021/05/28 05:24:59 [INFO] generate received request
   2021/05/28 05:24:59 [INFO] received CSR
   2021/05/28 05:24:59 [INFO] generating key: rsa-2048
   2021/05/28 05:24:59 [INFO] encoded CSR
   ```

3. Reinicie a implantação do webhook.

   ```
   kubectl rollout restart deployment -n kube-system vpc-admission-webhook
   ```

4. Se o certificado que você renovou tiver expirado e houver Pods do Windows presos no estado `Container creating`, será necessário excluir e reimplantar esses Pods.

Windows

1. Obtenha o script para gerar novo certificado.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.ps1;
   ```

2. Prepare o parâmetro para o script.

   ```
   ./webhook-create-signed-cert.ps1 -ServiceName vpc-admission-webhook-svc -SecretName vpc-admission-webhook-certs -Namespace kube-system
   ```

3. Reinicie a implantação do webhook.

```
kubectl rollout restart deployment -n kube-system vpc-admission-webhook-deployment
```

4. Se o certificado que você renovou tiver expirado e houver Pods do Windows presos no estado `Container creating`, será necessário excluir e reimplantar esses Pods.

Linux and macOS

Pré-requisito

Você deve ter o OpenSSL e o `jq` instalados em seu computador.

1. Obtenha o script para gerar novo certificado.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.sh
```

2. Altere as permissões.

```
chmod +x webhook-create-signed-cert.sh
```

3. Executar o script.

```
./webhook-create-signed-cert.sh
```

4. Reinicie o webhook.

```
kubectl rollout restart deployment -n kube-system vpc-admission-webhook-deployment
```

5. Se o certificado que você renovou tiver expirado e houver Pods do Windows presos no estado `Container creating`, será necessário excluir e reimplantar esses Pods.

Implementar clusters privados com acesso limitado à internet

Este tópico descreve como implantar um cluster do Amazon EKS implantado na Nuvem AWS, mas que não tem acesso de saída à Internet. Se você tiver um cluster local no AWS Outposts, consulte Criar nós Amazon Linux no AWS Outposts em vez deste tópico.

Se você não estiver familiarizado com a rede do Amazon EKS, consulte De-mystifying cluster networking for Amazon EKS worker nodes. Se o cluster não tiver acesso de saída à Internet, ele deverá atender aos seguintes requisitos:

- O cluster deve extrair imagens de um registro de contêiner que esteja na VPC. Você pode criar um Amazon Elastic Container Registry na VPC e copiar para ele as imagens de contêiner para que os nós as extrai dali. Para ter mais informações, consulte Copiar uma imagem de contêiner de um repositório para outro.
- O cluster deve ter acesso privado ao endpoint habilitado. Isso é necessário para que os nós sejam registrados no endpoint do cluster. O acesso público ao endpoint é opcional. Para ter mais informações, consulte Controlar o acesso à rede ao endpoint do servidor de API do cluster.
- Os nós autogerenciados do Linux e do Windows devem incluir os seguintes argumentos de bootstrap antes de serem iniciados. Esses argumentos ignoram a introspecção do Amazon EKS e não exigem acesso à API do Amazon EKS de dento da VPC.

  1. Determine o valor do endpoint do cluster com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

     ```
     aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
     ```

     Veja um exemplo de saída abaixo.

     ```
     https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
     ```

  2. Determine o valor da autoridade de certificação do cluster com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

     ```
     aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --output text
     ```

     A saída retornada é uma string longa.

  3. Substitua *cluster-endpoint* e *certificate-authority* nos comandos a seguir pelos valores retornados na saída dos comandos anteriores. Para obter mais informações sobre a especificação de argumentos de bootstrap ao iniciar nós autogerenciados, consulte Criar nós autogerenciados do Amazon Linux e Criar nós Microsoft Windows autogerenciados.

  - Para nós do Linux:

```
--apiserver-endpoint cluster-endpoint --b64-cluster-ca certificate-authority
```

Para obter argumentos adicionais, consulte bootstrap script (script de bootstrap) no GitHub.

- Para nós do Windows:

> **Note**
>
> Se você estiver usando um CIDR de serviço personalizado, será necessário especificá-lo usando o parâmetro `-ServiceCIDR`. Caso contrário, a resolução de DNS Pods no cluster falhará.

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

Para obter argumentos adicionais, consulte Parâmetros de configuração do script de bootstrap.

- O `ConfigMap` da `aws-auth` do cluster deve ser criado de dentro da VPC. Para obter mais informações sobre como criar e adicionar entradas ao `ConfigMap` `aws-auth`, insira **`eksctl create iamidentitymapping --help`** em seu terminal. Se o `ConfigMap` não existir no seu servidor, `eksctl` será criado quando você usar o comando para adicionar um mapeamento de identidade.
- Pods configurados com perfis do IAM para contas de serviço adquirem as credenciais com uma chamada de API do AWS Security Token Service (AWS STS). Se não houver acesso de saída à Internet, será necessário criar e usar um endpoint da VPC do AWS STS na VPC. A maior parte dos AWS `v1` SDKs usam o endpoint global do AWS STS por padrão (`sts.amazonaws.com`), que não usa o endpoint da VPC do AWS STS. Para usar o endpoint da VPC do AWS STS, pode ser necessário configurar o SDK para usar o endpoint do AWS STS regional (`sts.region-code.amazonaws.com`). Para ter mais informações, consulte Configurar o endpoint do AWS Security Token Service para uma conta de serviço.
- As sub-redes da VPC do cluster devem ter um endpoint da VPC de interface para todos os Serviços da AWS que os Pods precisam acessar. Para obter mais informações, consulte Acessar um serviço da AWS usando um endpoint da VPC de interface. Alguns serviços e endpoints comumente usados estão listados na tabela a seguir. Para obter uma lista completa de endpoints, consulte Serviços da AWS que se integram ao AWS PrivateLink no Guia do AWS PrivateLink.

Serviço	Endpoint
Amazon EC2	com.amazonaws.*region-code* .ec2
Amazon Elastic Container Registry (para extração de imagens de contêineres)	com.amazonaws.*region-code* .ecr.api, com.amazonaws.*region-code* .ecr.dkr e com.amazonaws.*region-code* .s3
Application Load Balancers e Network Load Balancers	com.amazonaws.*region-code* .elasticloadbalancing
AWS X-Ray	com.amazonaws.*cregion-code* .xray
Amazon CloudWatch Logs	com.amazonaws.*region-code* .logs
AWS Security Token Service (obrigatório quando estiver usando perfis do IAM para contas de serviço)	com.amazonaws.*region-code* .sts

Considerações

- Todos os nós autogerenciados devem ser implantados em sub-redes que tenham os endpoints de interface da VPC necessários. Se você criar um grupo de nós gerenciados, o grupo de segurança de endpoint de interface da VPC deverá aceitar o CIDR para as sub-redes ou será necessário adicionar o grupo de segurança do nó criado ao grupo de segurança do endpoint de interface da VPC.
- Se os Pods usarem volumes do Amazon EFS, antes de implantar o Armazenar um sistema de arquivos elástico com o Amazon EFS, será necessário alterar o arquivo kustomization.yaml para definir as imagens de contêiner a serem usadas na mesma Região da AWS do cluster do Amazon EKS.
- Você pode usar o AWS Load Balancer Controller para implantar Application Load Balancers (ALBs) and Network Load Balancers da AWS no cluster privado. Ao implantá-lo, você deve usar sinalizadores de linha de comando para definir `enable-shield`, `enable-waf` e `enable-wafv2` como falsos. Detecção de certificados com nomes de host de objetos de ingresso não é compatível. Isso porque o controlador precisa alcançar o AWS Certificate Manager, que não tem um endpoint de interface da VPC.

O controlador é compatível com balanceadores de carga de rede com destinos IP, que são necessários para uso com Fargate. Para ter mais informações, consulte Aplicação de roteamento e tráfego HTTP com Application Load Balancers e Criar um balanceador de carga da rede.

- Há suporte para o Cluster Autoscaler. Ao implantar Pods do Cluster Autoscaler, certifique-se de que a linha de comando inclua `--aws-use-static-instance-list=true`. Para obter mais informações, consulte Use Static Instance List (Usar lista de instâncias estáticas) no GitHub. A VPC do nó de processamento também inclui o endpoint da VPC do AWS STS e o endpoint da VPC de escalação automática.
- Alguns produtos de software de contêiner usam chamadas de API que acessam o AWS Marketplace Metering Service para monitorar sua utilização. Como os clusters privados não permitem essas chamadas, esses tipos de contêiner não podem ser usados nesses clusters.

Compreeender o ciclo de vida da versão do Kubernetes no EKS

Kubernetes evolui rapidamente com novos recursos, atualizações de design e correções de bugs. A comunidade lança novas versões Kubernetes secundárias (como `1.30`) em média uma vez a cada quatro meses. O Amazon EKS segue o ciclo upstream de lançamento e descontinuação para versões secundárias. Conforme novas versões do Kubernetes são disponibilizadas no Amazon EKS, recomendamos que você atualize proativamente seus clusters para usarem a versão mais recente disponível.

Uma versão secundária terá suporte padrão no Amazon EKS durante os primeiros 14 meses após seu lançamento. Depois que uma versão excede a data de término do suporte padrão, ela entra no suporte estendido pelos 12 meses seguintes. O suporte estendido permite que você permaneça em uma Kubernetes versão específica por mais tempo por um custo adicional por hora de cluster. Se você não atualizou seu cluster antes do término do período de suporte estendido, seu cluster é atualizado automaticamente para a versão estendida mais antiga atualmente suportada.

O suporte estendido está habilitado por padrão. É possível desabilitar o suporte estendido.

Recomendamos que você crie seu cluster com a versão Kubernetes mais recente disponível e compatível com o Amazon EKS.. Se seu aplicativo exigir uma versão específica do Kubernetes, você poderá selecionar versões mais antigas. Você pode criar novos clusters do Amazon EKS em qualquer versão oferecida no suporte padrão ou estendido.

Versões disponíveis no suporte padrão

Atualmente, as seguintes Kubernetes versões estão disponíveis no suporte padrão do Amazon EKS:

- `1.30`
- `1.29`
- `1.28`

Para conhecer alterações importantes para cada versão no suporte padrão, consulte Revisão das notas de release das versões do Kubernetes com suporte padrão.

Versões disponíveis com suporte estendido

Atualmente, as seguintes versões Kubernetes estão disponíveis no suporte estendido do Amazon EKS:

- `1.27`
- `1.26`
- `1.25`
- `1.24`
- `1.23`

Para conhecer alterações importantes para cada versão no suporte estendido, consulte Revisão das notas de release das versões do Kubernetes com suporte estendido.

As seguintes versões do Kubernetes estão disponíveis no suporte estendido do Amazon EKS, com a exigência adicional de que não é possível criar novos clusters com essas versões:

- `1.22`

Para obter mais informações sobre essas versões, consulte Revisão das notas de versão do Kubernetes versão 1.22.

Calendário de lançamento do Amazon EKS Kubernetes

A tabela a seguir mostra datas importantes de lançamento e suporte a serem consideradas para cada versão Kubernetes. A cobrança do suporte estendido começa no início do dia em que a versão atinge o final do suporte padrão.

> **Note**
>
> As datas com apenas mês e ano são aproximadas e são atualizadas com uma data exata quando ela é conhecida.

Versão do Kubernetes	Versão upstream	Lançamento do Amazon EKS	Fim do suporte padrão	Fim do suporte estendido
1.30	17 de abril de 2024	23 de maio de 2024	23 de julho de 2025	23 de julho de 2026
1.29	13 de dezembro de 2023	23 de janeiro de 2024	23 de março de 2025	23 de março de 2026
1.28	15 de agosto de 2023	26 de setembro de 2023	26 de novembro de 2024	26 de novembro de 2025
1.27	11 de abril de 2023	24 de maio de 2023	24 de julho de 2024	24 de julho de 2025
1.26	9 de dezembro de 2022	11 de abril de 2023	11 de junho de 2024	11 de junho de 2025
1.25	23 de agosto de 2022	22 de fevereiro de 2023	1º de maio de 2024	1.º de maio de 2025
1.24	3 de maio de 2022	15 de novembro de 2022	31 de janeiro de 2024	31 de janeiro de 2025
1.23	7 de dezembro de 2021	11 de agosto de 2022	11 de outubro de 2023	11 de outubro de 2024

Versão do Kubernetes	Versão upstream	Lançamento do Amazon EKS	Fim do suporte padrão	Fim do suporte estendido
1.22	4 de agosto de 2021	4 de abril de 2022	4 de junho de 2023	1º de setembro de 2024
1.21	8 de abril de 2021	19 de julho de 2021	16 de fevereiro de 2023	15 de julho de 2024

Perguntas frequentes da versão do Amazon EKS

Quantas versões Kubernetes estão disponíveis no suporte padrão?

> De acordo com o suporte da comunidade do Kubernetes para versões do Kubernetes, o Amazon EKS está comprometido em oferecer suporte estendido a pelo menos quatro versões do Kubernetes prontas para produção a qualquer momento. Anunciaremos a data de encerramento do suporte padrão de determinada versão Kubernetes secundária, com pelo menos 60 dias de antecedência. Devido ao processo de lançamento e qualificação do Amazon EKS para novas versões do Kubernetes, a data de encerramento padrão de uma versão do Kubernetes no Amazon EKS será a data em que o projeto do Kubernetes interrompe o suporte ao upstream da versão ou após essa data.

Por quanto tempo um Kubernetes recebe suporte padrão do Amazon EKS?

> Uma versão do Kubernetes recebeu suporte padrão por 14 meses após estar disponível pela primeira vez no Amazon EKS. Isso também será válido mesmo se o Kubernetes upstream não for mais compatível com uma versão disponível no Amazon EKS. Fornecemos patches de segurança de backport que são aplicáveis às versões do Kubernetes compatíveis com o Amazon EKS.

Sou notificado quando o suporte estiver terminando para uma versão do Kubernetes no Amazon EKS?

> Sim. Se algum cluster em sua conta estiver executando a versão que está se aproximando do fim do suporte, o Amazon EKS enviará um aviso pelo AWS Health Dashboard aproximadamente 12 meses após a versão do Kubernetes ter sido lançada no Amazon EKS. O aviso inclui a data de fim do suporte. Isso é, pelo menos, 60 dias após a data do aviso.

Quais recursos do Kubernetes são compatíveis com o Amazon EKS?

O Amazon EKS oferece suporte a todos os recursos de disponibilidade geral (GA) do Kubernetes. A partir do Kubernetes versão `1.24`, as novas APIs beta não estão habilitadas nos clusters por padrão. As APIs beta anteriores existentes e as novas versões das APIs beta existentes continuam a ser habilitadas por padrão. Não há suporte a recursos alfa.

Os grupos de nós gerenciados pelo Amazon EKS são atualizados automaticamente junto com a versão do ambiente de gerenciamento de cluster?

Não. Um grupo de nós gerenciados cria instâncias do Amazon EC2 em sua conta. Essas instâncias não são atualizadas automaticamente quando você ou o Amazon EKS atualizam seu plano de controle. Para obter mais informações, consulte Atualizar um grupo de nós gerenciados para seu cluster. Recomendamos manter a mesma versão do Kubernetes em seu ambiente de gerenciamento e nos nós.

P: Os grupos de nós autogerenciados são atualizados automaticamente junto com a versão do ambiente de gerenciamento de cluster?

Não. Um grupo de nós autogerenciado inclui instâncias do Amazon EC2 em sua conta. Essas instâncias não são atualizadas automaticamente quando você ou o Amazon EKS atualizam a versão do seu ambiente de gerenciamento em seu nome. Um grupo de nós autogerenciado não tem nenhuma indicação no console de que ele precisa ser atualizado. Você pode visualizar a versão do `kubelet` instalada em um nó selecionando o nó na lista Nodes (Nós) da guia Overview (Visão geral) do cluster para determinar quais nós precisam ser atualizados. Você deve atualizar manualmente os nós. Para obter mais informações, consulte Atualizar nós autogerenciados para seu cluster.

O projeto Kubernetes testa a compatibilidade entre o ambiente de gerenciamento e os nós para até três versões secundárias. Por exemplo, os nós `1.27` continuam a operar quando orquestrados por um ambiente de gerenciamento `1.30`. No entanto, a execução de um cluster com nós que são persistentemente três versões secundárias atrás do ambiente de gerenciamento não é recomendada. Para obter mais informações, consulte a Política de suporte da distorção da versão e da versão do Kubernetes, na documentação do Kubernetes. Recomendamos manter a mesma versão do Kubernetes em seu ambiente de gerenciamento e nos nós.

Os Pods em execução no Fargate são atualizados automaticamente com uma atualização automática de versão do ambiente de gerenciamento do cluster?

Não. É extremamente recomendável executar os Pods do Fargate como parte de um controlador de replicação, como uma implantação do Kubernetes. Em seguida, faça uma reinicialização sucessiva de todos os Pods do Fargate. A nova versão do Pod Fargate é implantada com uma versão do `kubelet` que é a mesma da versão atualizada do ambiente de gerenciamento do cluster. Para obter mais informações, consulte Implantações na documentação do Kubernetes.

> **Important**
>
> Se você atualizar o ambiente de gerenciamento, ainda será necessário atualizar os nós Fargate por conta própria. Para atualizar os nós do Fargate, exclua o Pod Fargate representado pelo nó e reimplante o Pod. O novo Pod é implantado com uma versão do `kubelet` que é a mesma versão do cluster.

Perguntas frequentes sobre o suporte estendido do Amazon EKS

A terminologia padrão de suporte e suporte estendido é nova para mim. O que significam esses termos?

O suporte padrão para uma versão Kubernetes no Amazon EKS começa quando uma versão Kubernetes é lançada no Amazon EKS e terminará 14 meses após a data de lançamento. O suporte estendido para uma versão Kubernetes começará imediatamente após o término do suporte padrão e terminará após os próximos 12 meses. Por exemplo, o suporte padrão para a versão `1.23` no Amazon EKS termina em 11 de outubro de 2023. O suporte estendido para a versão `1.23` começou em 12 de outubro de 2023 e terminará em 11 de outubro de 2024.

O que eu preciso fazer para obter suporte estendido para clusters do Amazon EKS?

Você não precisa realizar nenhuma ação para obter suporte estendido para seus clusters do Amazon EKS. O suporte padrão começará quando uma versão Kubernetes for lançada no Amazon EKS e terminará 14 meses após a data de lançamento. O suporte estendido para uma versão Kubernetes começará imediatamente após o término do suporte padrão e terminará após os próximos 12 meses.

Para quais versões Kubernetes eu posso obter suporte estendido?

O suporte estendido está disponível para Kubernetes versões `1.23` e superiores. Você pode executar clusters em qualquer versão por até 12 meses após o fim do suporte padrão para essa versão. Isso significa que cada versão terá suporte por 26 meses no Amazon EKS (14 meses de suporte padrão mais 12 meses de suporte estendido).

E se eu não quiser usar o suporte estendido?

Se você não quiser se inscrever automaticamente no suporte estendido, você pode atualizar seu cluster para uma versão Kubernetes são que esteja no suporte padrão do Amazon EKS. Também é possível desabilitar o suporte estendido.

O que acontecerá ao final de 12 meses de suporte estendido?

Os clusters executados em uma versão Kubernetes que completou seu ciclo de vida de 26 meses (14 meses de suporte padrão mais 12 meses de suporte estendido) serão atualizados automaticamente para a próxima versão.

Na data de fim do suporte estendido, não será mais possível criar novos clusters do Amazon EKS com a versão sem suporte. Os planos de controle existentes são atualizados automaticamente pelo Amazon EKS para a versão mais antiga com suporte por meio de um processo de implantação gradual após a data de término do suporte. Após a atualização automática do ambiente de gerenciamento, certifique-se de atualizar manualmente os complementos do cluster e os nós do Amazon EC2. Para obter mais informações, consulte Atualizar a versão do Kubernetes de um cluster do Amazon EKS.

Quando exatamente meu ambiente de gerenciamento será atualizado automaticamente após a data do fim do suporte estendido?

O Amazon EKS não pode fornecer prazos específicos. Atualizações automáticas podem ocorrer a qualquer momento após a data de fim do suporte estendido. Você não receberá nenhuma notificação antes da atualização. Recomendamos que você atualize proativamente seu ambiente de gerenciamento sem depender do processo de atualização automática do Amazon EKS. Para obter mais informações, consulte Atualizar um cluster existente para a nova versão do Kubernetes.

Posso deixar meu ambiente de gerenciamento em uma versão do Kubernetes indefinidamente?

Não, segurança da nuvem na AWS é a nossa maior prioridade. Após um determinado ponto (geralmente um ano), a comunidade do Kubernetes para de lançar patches para common

vulnerabilities and exposures (CVE – Vulnerabilidades e exposições comuns) e desencoraja o envio de CVEs para versões incompatíveis. Isso quer dizer que vulnerabilidades específicas de uma versão anterior doKubernetes podem nem mesmo ter sido relatadas. Isso deixa clusters expostos sem aviso prévio e sem opções de correção em caso de vulnerabilidade. Considerando isso, o Amazon EKS não permite que ambientes de gerenciamento permaneçam em uma versão que tenha atingido o fim do suporte estendido.

Há algum custo adicional para obter suporte estendido?

Sim, haverá um custo adicional para clusters do Amazon EKS executados em suporte estendido. Para obter detalhes sobre preços, consulte o Amazon EKS extended support for Kubernetes version pricing no blog da AWS.

O que está incluído no suporte estendido?

Os clusters do Amazon EKS no Extended Support recebem patches de segurança contínuos para oambiente de gerenciamento Kubernetes. Além disso, o Amazon EKS lançará patches para o Amazon VPC CNI e CoreDNS complementos para as versões do suprote estendido `kube-proxy`. O Amazon EKS também lançará patches para AMIs otimizadas do Amazon EKS AWS publicadas para Amazon Linux, Bottlerocket e Windows, bem como nós Fargate do Amazon EKS para essas versões. Todos os clusters do suporte estendido continuarão a ter acesso ao suporte técnico do AWS.

> **Note**
>
> O Extended Support para AMIs do Windows otimizadas do Amazon EKS que são publicadas pela AWS não está disponível para o Kubernetes versão`1.23`, mas está disponível para o Kubernetes versão `1.24` e superior.

Há alguma limitação nos patches para componentes que não Kubernetes no suporte estendido?

Embora o Extended Support cubra todos os componentes Kubernetes específicos da AWS, ele fornecerá suporte apenas para AMIs otimizadas do Amazon EKS AWS publicadas para Amazon Linux, Bottlerocket e Windows em todos os momentos. Isso significa que você potencialmente terá componentes mais novos (como sistema operacional ou kernel) em sua AMI otimizada para Amazon EKS enquanto estiver usando o suporte estendido. Por exemplo, quando o Amazon Linux 2 chegar ao fim de seu ciclo de vida em 2025, as AMIs do Amazon Linux otimizadas para Amazon EKS serão criadas usando um sistema operacional Amazon Linux mais novo. O Amazon

EKS anunciará e documentará discrepâncias importantes no ciclo de vida do suporte, como essa, para cada versão. Kubernetes

Posso criar novos clusters usando uma versão com suporte estendido?

Sim, exceto com as versões `1.22` e `1.21`. Por exemplo, você pode criar um cluster `1.23`, mas não um cluster `1.22`.

Revisão das notas de release das versões do Kubernetes com suporte padrão

Este tópico fornece mudanças importantes que você deve conhecer em cada Kubernetes versão do suporte padrão. Ao fazer o upgrade, analise cuidadosamente as alterações que ocorreram entre a versão antiga e a nova do seu cluster.

> **Note**
>
> Para clusters `1.24` e posteriores, as AMIs do Amazon EKS publicadas oficialmente incluem o `containerd` como o único runtime. As versões do Kubernetes anteriores à `1.24` usam o Docker como runtime padrão. Essas versões têm uma opção de sinalizador de bootstrap que você pode usar para testar as workloads em qualquer cluster compatível com o `containerd`. Para obter mais informações, consulte Migrar de `dockershim` para `containerd`.

Kubernetes 1.30

O Kubernetes `1.30` já está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.30`, consulte o anúncio oficial de lançamento.

> **Important**
>
> - A partir da versão `1.30` do Amazon EKS ou em versões mais recentes, qualquer grupo de nós gerenciados recém-criados será automaticamente padronizado para usar o Amazon Linux 2023 (AL2023) como o sistema operacional do nó. Anteriormente, os novos grupos de nós seriam padronizados para usar o Amazon Linux 2 (AL2). É possível continuar a usar AL2 ao escolhê-lo como o tipo de AMI durante a criação de um novo grupo de nós.

- Para obter mais informações sobre o Amazon Linux, consulte Comparação entre o AL2 e o AL2023, no Guia do Usuário do Amazon Linux.
- Para obter mais informações sobre como especificar o sistema operacional para um grupo de nós gerenciados, consulte Criar um grupo de nós gerenciados para seu cluster.

- Com o Amazon EKS `1.30`, o rótulo `topology.k8s.aws/zone-id` é adicionado a nós de processamento. É possível visualizar os IDs de Zonas de disponibilidade (IDs de AZs) para determinar o local de recursos em uma conta em relação aos recursos em outra conta. Para obter mais informações, consulte IDs de zonas de disponibilidade para seus recursos da AWS, no Guia do usuário do AWS RAM.
- A partir da versão `1.30`, o Amazon EKS não incluirá mais a anotação `default` no recurso `gp2` `StorageClass` aplicado a clusters recém-criados. Isso não terá impacto se você estiver referenciando essa classe de armazenamento pelo nome. Você deverá tomar medidas se estiver confiando em ter um `StorageClass` padrão no cluster. É necessário referenciar `StorageClass` pelo nome `gp2`. Como alternativa, você pode implantar a classe de armazenamento padrão recomendada pelo Amazon EBS definindo o parâmetro `defaultStorageClass.enabled` como verdadeiro ao instalar a `v1.31.0` ou posterior do `aws-ebs-csi-driver add-on`.
- A política do IAM mínima exigida para o perfil do IAM do cluster do Amazon EKS foi alterada. A ação `ec2:DescribeAvailabilityZones` é obrigatória. Para obter mais informações, consulte Função do IAM do cluster do Amazon EKS.

Para ver o log de alterações completo do Kubernetes `1.30`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.30.md.

Kubernetes 1.29

O Kubernetes `1.29` já está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.29`, consulte o anúncio oficial de lançamento.

Important

- A versão obsoleta da API `flowcontrol.apiserver.k8s.io/v1beta2` do `FlowSchema` e `PriorityLevelConfiguration` foram descontinuados e não são mais

exibidas no Kubernetes `v1.29`. Se você tiver manifestos ou software cliente que usa o grupo de API beta obsoleto, altere-os antes de fazer o upgrade para a versão `v1.29`.

- O campo `.status.kubeProxyVersion` para objetos de nó agora está obsoleto, e o projeto Kubernetes está propondo remover esse campo em uma versão futura. O campo obsoleto não é preciso e historicamente foi gerenciado por `kubelet` que, na verdade, não conhece a versão `kube-proxy` ou mesmo se `kube-proxy` está em execução. Se você estiver usando este campo no software cliente, pare. As informações não são confiáveis e o campo agora está obsoleto.
- No Kubernetes `1.29`, para reduzir a potencial superfície de ataque, o recurso `LegacyServiceAccountTokenCleanUp` rotula os tokens legados baseados em segredos gerados automaticamente como inválidos se não forem usados por um longo período (1 ano por padrão) e os remove automaticamente se o uso não for tentado por um longo período após serem marcados como inválidos (1 ano adicional por padrão). Para identificar esses tokens, você pode executar:

```
kubectl get cm kube-apiserver-legacy-service-account-token-tracking -n kube-system
```

Para ver o log de alterações completo do Kubernetes `1.29`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.29.md#changelog-since-v1280.

Kubernetes 1.28

O Kubernetes `1.28` já está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.28`, consulte o anúncio oficial de lançamento.

- Kubernetes `v1.28` expandiu a inclinação suportada entre o nó central e os componentes do ambiente de gerenciamento em uma versão secundária, de `n-2` para `n-3`, para que os componentes do nó (`kubelet` e `kube-proxy`) da versão secundária suportada mais antiga possam funcionar com os componentes do ambiente de gerenciamento (`kube-apiserver`, `kube-scheduler`, `kube-controller-manager`, `cloud-controller-manager`) para a versão secundária suportada mais recente.
- As métricas `force_delete_pods_total` e `force_delete_pod_errors_total` no `Pod GC Controller` são aprimoradas para considerar a exclusão forçada de todos os pods. Um motivo é adicionado à métrica para indicar se o pod está sendo excluído por força, porque está sendo encerrado, órfão, encerrando com a taint "out-of-service" ou encerrando e não agendado.

- O `PersistentVolume (PV)` controlador foi modificado para atribuir automaticamente um padrão `StorageClass` a qualquer não vinculado `PersistentVolumeClaim` com o `storageClassName` não definido. Além disso, o mecanismo de validação de `PersistentVolumeClaim` admissão no servidor da API foi ajustado para permitir a alteração de valores de um estado não definido para um `StorageClass` nome real.

Para ver o log de alterações completo do Kubernetes `1.28`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.28.md#changelog-since-v1270.

Revisão das notas de release das versões do Kubernetes com suporte estendido

Este tópico fornece mudanças importantes que você deve conhecer em cada Kubernetes versão do suporte estendido. Ao fazer o upgrade, analise cuidadosamente as alterações que ocorreram entre a versão antiga e a nova do seu cluster.

Kubernetes 1.27

O Kubernetes `1.27` já está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.27`, consulte o anúncio oficial de lançamento.

Important

- O suporte para anotações `seccomp.security.alpha.kubernetes.io/pod` e `seccomp container.seccomp.security.alpha.kubernetes.io` anotações alfa foi removido. As `seccomp` anotações alfa foram descontinuadas em `1.19`, e com sua remoção em `1.27`, os `seccomp` campos não serão mais preenchidos automaticamente para `seccomp` com `Pods` anotações. Em vez disso, use o `securityContext.seccompProfile` campo para `Pods` ou contêineres para configurar `seccomp` perfis. Para verificar se você está usando as anotações alpha `seccomp` obsoletas em seu cluster, execute o seguinte comando:

  ```
  kubectl get pods --all-namespaces -o json | grep
   -E 'seccomp.security.alpha.kubernetes.io/pod|
  container.seccomp.security.alpha.kubernetes.io'
  ```

- O argumento da linha de `--container-runtime` comando para o `kubelet` foi removido. O runtime padrão do contêiner para Amazon EKS é `containerd` desde a

versão 1.24, o que elimina a necessidade de especificar o runtime do contêiner. A partir de 1.27 em diante, o Amazon EKS ignorará o --container-runtime argumento passado para qualquer script de bootstrap. É importante que você não passe esse argumento para --kubelet-extra-args evitar erros durante o processo de bootstrap do nó. Você deve remover o --container-runtime argumento de todos os seus fluxos de trabalho de criação de nós e criar scripts.

- O kubelet em Kubernetes 1.27 aumentou o padrão de kubeAPIQPS para 50 e de kubeAPIBurst para 100. Esses aprimoramentos permitem kubelet lidar com um volume maior de consultas de API, melhorando os tempos de resposta e o desempenho. Quando as demandas de escalonamento Pods aumentam, devido aos requisitos de escalabilidade, os padrões revisados garantem que eles kubelet possam gerenciar com eficiência o aumento do workload. Como resultado, os Pod lançamentos são mais rápidos e as operações de cluster são mais eficazes.
- Você pode usar uma Pod topologia mais refinada para difundir políticas como minDomain. Esse parâmetro permite especificar o número mínimo de domínios pelos quais você Pods deve estar distribuído. nodeAffinityPolicy e nodeTaintPolicy permita um nível extra de granularidade na Pod governança da distribuição. Isso está de acordo com as afinidades dos nós, as manchas e o matchLabelKeys campo topologySpreadConstraints em sua especificação Pod's. Isso permite a seleção de cálculos Pods para distribuição após uma atualização contínua.
- Kubernetes1.27 promoveu à versão beta um novo mecanismo de política StatefulSets que controla a vida útil de seu PersistentVolumeClaims (PVCs). A nova política PVC de retenção permite especificar se o PVCs gerado a partir do modelo de StatefulSet especificação será automaticamente excluído ou retido quando StatefulSet for excluído ou se as réplicas StatefulSet forem reduzidas.
- Por meio do fechamento aleatório de uma conexão, a opção goaway-chance no servidor de API do Kubernetes ajuda a evitar que as conexões do cliente HTTP/2 fiquem presas em uma única instância do servidor de API. Quando a conexão for fechada, o cliente tentará se reconectar e provavelmente acessará um servidor de API diferente como resultado do balanceamento de carga. O Amazon EKS versão 1.27 tem a sinalização goaway-chance habilitada. Se sua workload em execução no cluster do Amazon EKS usa um cliente que não é compatível com HTTP GOAWAY, recomendamos atualizar seu cliente para lidar com GOAWAY reconectando-se no encerramento da conexão.

Para ver o log de alterações completo do Kubernetes 1.27, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.27.md#changelog-since-v1260.

Kubernetes 1.26

O Kubernetes 1.26 já está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes 1.26, consulte o anúncio oficial de lançamento.

> **Important**
>
> O Kubernetes 1.26 não oferece mais suporte a CRI `v1alpha2`. Isso faz com que o `kubelet` deixe de registrar o nó se o runtime do contêiner não for compatível com o CRI `v1`. Isso também significa que o Kubernetes 1.26 não oferece suporte à versão secundária 1.5 do containerd e versões anteriores. Se você estiver usando containerd, precisará atualizar para o containerd versão 1.6.0 ou posterior antes de atualizar qualquer nó para o Kubernetes 1.26. Você também precisa atualizar qualquer outro runtime de contêiner que seja compatível apenas com o `v1alpha2`. Para obter mais informações, consulte o fornecedor de runtime de contêiner. Por padrão, as AMIs Amazon Linux e Bottlerocket incluem o containerd versão 1.6.6.

- Antes de fazer o upgrade para Kubernetes 1.26, atualize seu Amazon VPC CNI plugin for Kubernetes para a versão 1.12 ou posterior. Se você não fizer o upgrade para o Amazon VPC CNI plugin for Kubernetes versão 1.12 ou posterior, o Amazon VPC CNI plugin for Kubernetes vai falhar. Para obter mais informações, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.
- Por meio do fechamento aleatório de uma conexão, a opção `goaway-chance` no servidor de API do Kubernetes ajuda a evitar que as conexões do cliente HTTP/2 fiquem presas em uma única instância do servidor de API. Quando a conexão for fechada, o cliente tentará se reconectar e provavelmente acessará um servidor de API diferente como resultado do balanceamento de carga. O Amazon EKS versão 1.26 tem a sinalização `goaway-chance` habilitada. Se sua workload em execução no cluster do Amazon EKS usa um cliente que não é compatível com `HTTP GOAWAY`, recomendamos atualizar seu cliente para lidar com `GOAWAY` reconectando-se no encerramento da conexão.

Para ver o log de alterações completo do Kubernetes 1.26, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.26.md#changelog-since-v1250.

Kubernetes 1.25

O Kubernetes `1.25` já está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.25`, consulte o anúncio oficial de lançamento.

Important

- A partir do Kubernetes versão `1.25`, não será mais possível usar instâncias `P2` do Amazon EC2 com as AMIs aceleradas do Amazon Linux otimizadas para o Amazon EKS prontas para uso. Essas AMIs para o Kubernetes versões `1.25` ou posteriores serão compatíveis com drivers série `NVIDIA 525` ou posterior, que são incompatíveis com as instâncias `P2`. No entanto, os drivers da série `NVIDIA 525` ou posteriores são compatíveis com as instâncias `P3`, `P4` e `P5`. Portanto, é possível usar essas instâncias com as AMIs para o Kubernetes versão `1.25` ou posterior. Antes que seus clusters do Amazon EKS sejam atualizados para a versão `1.25`, migre qualquer instância `P2` para instâncias `P3`, `P4` e `P5`. Você também deverá atualizar proativamente suas aplicações para funcionar com a série `NVIDIA 525` ou posterior. Planejamos transferir os drivers da série `NVIDIA 525` ou mais recente ou para as versões `1.23` e `1.24` do Kubernetes no final de janeiro de 2024.
- A `PodSecurityPolicy` (PSP) é removida em Kubernetes `1.25`. PSPs são substituídos pelos Pod Security Admission (PSA) e pelos (PSS) do Pod Security Standards. PSA é um controlador de admissão integrado que implementa os controles de segurança descritos no PSS. PSA e PSS são graduados para estabilizar em Kubernetes `1.25` e estão habilitados no Amazon EKS por padrão. Se você tiver PSPs no cluster, migre de PSP para a solução integrada Kubernetes PSS ou para uma solução de política como código antes de atualizar o cluster para a versão `1.25`. Se você não migrar do PSP, poderá encontrar interrupções em suas workloads. Para obter mais informações, consulte Perguntas frequentes sobre a remoção da política de segurança do Pod (PSP).
- O Kubernetes versão `1.25` contém mudanças que alteram o comportamento de um recurso existente conhecido como APF (API Priority and Fairness – Prioridade e equidade de API). O APF serve para proteger o servidor de API contra uma possível sobrecarga durante períodos de maior volume de solicitações. Para fazer isso, o recurso impõe restrições ao número de solicitações simultâneas passíveis de processamento a qualquer momento. Esse comportamento é alcançado com a aplicação de níveis e limites de prioridade distintos às solicitações provenientes de várias workloads ou usuários. Essa abordagem garante que aplicações críticas ou solicitações de alta prioridade recebam tratamento preferencial enquanto evita que solicitações de baixa prioridade

sobrecarreguem o servidor da API. Para obter mais informações, consulte Prioridade e equidade de API na documentação do Kubernetes ou Prioridade e equidade de API no Guia de melhores práticas do EKS.

Essas atualizações foram introduzidas nas PR n.º 10352 e PR n.º 118601. Anteriormente, a APF tratava todos os tipos de solicitações de maneira uniforme, com cada solicitação consumindo uma unidade do limite de solicitações simultâneas. A mudança de comportamento do APF atribui unidades superiores de simultaneidade a solicitações `LIST` devido à carga excepcionalmente pesada imposta ao servidor de API por essas solicitações. O servidor de API faz a estimativa do número de objetos que serão retornados por um pedido `LIST`. Ele atribui uma unidade de simultaneidade que é proporcional ao número de objetos retornados.

Ao atualizar para a versão `1.25` ou superior do Amazon EKS, esse comportamento atualizado pode fazer com que workloads contendo solicitações `LIST` pesadas (que antes funcionavam sem problemas) se deparem com limites de taxa. Isso seria indicado por um código de resposta HTTP 429. Para evitar possíveis interrupções na workload devido à limitação de taxa para solicitações `LIST`, recomendamos veementemente que você reestruture suas workloads a fim de reduzir a incidência dessas solicitações. Como alternativa, você pode resolver esse problema ajustando as configurações do APF para alocar mais capacidade para solicitações essenciais enquanto reduz a capacidade alocada para solicitações não essenciais. Para obter mais informações sobre essas técnicas de mitigação, consulte Como prevenir o descarte de solicitações no Guia de melhores práticas do EKS.

- O Amazon EKS `1.25` inclui aprimoramentos na autenticação de cluster que contêm YAML bibliotecas atualizadas. Se um valor YAML no `aws-auth ConfigMap` encontrado no namespace `kube-system` começar com uma macro, em que o primeiro caractere for uma chave, você deve adicionar aspas (`" "`) antes e depois das chaves (`{ }`). Isso é necessário para garantir que o `aws-iam-authenticator` versão `v0.6.3` analise com precisão o `aws-auth ConfigMap` no Amazon EKS `1.25`.
- A versão beta da API (`discovery.k8s.io/v1beta1`) do `EndpointSlice` foi descontinuada no Kubernetes `1.21` e não é mais exibida a partir do Kubernetes `1.25`. Essa API foi atualizada para `discovery.k8s.io/v1`. Para obter mais informações, consulte a `EndpointSlice` documentação do Kubernetes. O AWS Load Balancer Controller `v2.4.6` e anteriores usavam o endpoint `v1beta1` para se comunicar com o `EndpointSlices`. Se você estiver usando a configuração `EndpointSlices` do AWS Load Balancer Controller, deverá atualizá-la para AWS Load Balancer Controller `v2.4.7`

antes de atualizar o cluster do Amazon EKS para `1.25`. Se você fizer o upgrade para `1.25` enquanto estiver usando a configuração `EndpointSlices` do AWS Load Balancer Controller, o controlador falhará e resultará em interrupções nas workloads. Para atualizar o controlador, consulte O que é o AWS Load Balancer Controller?.

- A versão beta da API (`autoscaling/v2beta1`) do HorizontalPodAutoscaler não é mais fornecida a partir do Kubernetes `1.25`. Essa API foi descontinuada na versão `1.23`. Migre manifestos e clientes de API para usar a versão da API HorizontalPodAutoscaler do `autoscaling/v2`. Para obter mais informações, consulte a documentação do Kubernetes.

- `SeccompDefault` é promovido à versão beta em Kubernetes `1.25`. Ao definir o sinalizador `--seccomp-default` quando você configura `kubelet`, o runtime do contêiner usa o perfil `RuntimeDefaultseccomp`, em vez do modo ‘’não confinado’’(`seccomp disabled`). Os perfis padrão fornecem um conjunto robusto de padrões de segurança, preservando a funcionalidade da workload. Embora esse sinalizador esteja disponível, o Amazon EKS não habilita esse sinalizador por padrão, portanto, o comportamento do Amazon EKS permanece efetivamente inalterado. Se quiser, você pode começar a habilitar isso em seus nós. Para obter mais detalhes, consulte o tutorial Restrict a Container's Syscalls with seccomp (Restringir as Syscalls de um contêiner com seccomp) na documentação do Kubernetes.
- A compatibilidade com a Container Runtime Interface (CRI) para Docker (também conhecida como `dockershim`) foi removida do Kubernetes `1.24` e posterior. O único runtime de contêineres nas AMIs oficiais do Amazon EKS para Kubernetes `1.24` e clusters posteriores é `containerd`. Antes de fazer o upgrade para o Amazon EKS `1.24` ou posterior, remova qualquer referência aos sinalizadores de script de bootstrap que não forem mais compatíveis. Para obter mais informações, consulte Migrar de `dockershim` para `containerd`.
- O suporte para consultas curinga foi descontinuado no CoreDNS `1.8.7` e removido no CoreDNS `1.9`. Isso foi feito como medida de segurança. As consultas curinga não funcionam mais e retornam `NXDOMAIN` em vez de um endereço IP.
- Por meio do fechamento aleatório de uma conexão, a opção `goaway-chance` no servidor de API do Kubernetes ajuda a evitar que as conexões do cliente `HTTP/2` fiquem presas em uma única instância do servidor de API. Quando a conexão for fechada, o cliente tentará se reconectar e provavelmente acessará um servidor de API diferente como resultado do balanceamento de carga. O Amazon EKS versão `1.25` tem a sinalização `goaway-chance` habilitada. Se sua workload em execução no cluster do Amazon EKS usa um cliente que não é compatível com `HTTP GOAWAY`,

recomendamos atualizar seu cliente para lidar com `GOAWAY` reconectando-se no encerramento da conexão.

Para ver o log de alterações completo do Kubernetes `1.25`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.25.md#changelog-since-v1240.

Kubernetes 1.24

O Kubernetes `1.24` já está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.24`, consulte o anúncio oficial de lançamento.

Important

- A partir do Kubernetes `1.24`, as novas APIs beta não estão habilitadas nos clusters por padrão. Por padrão, as APIs beta existentes e as novas versões das APIs beta existentes continuam a ser habilitadas. O Amazon EKS segue o mesmo comportamento que o Kubernetes `1.24` upstream. As portas de recursos que controlam novos recursos para operações de API novas e existentes são habilitadas por padrão. Isso está alinhado com o Kubernetes upstream. Para obter mais informações, consulte KEP-3136: Beta APIs Are Off by Default (KEP-3136: as APIs beta sã desativadas por padrão) no GitHub.
- A compatibilidade com a Container Runtime Interface (CRI) para Docker (também conhecida como `dockershim`) foi removida do Kubernetes `1.24`. As AMIs oficiais do Amazon EKS têm o containerd como único runtime. Antes de mudar para o Amazon EKS `1.24` ou superior, você deverá remover qualquer referência aos sinalizadores de script de bootstrap que não forem mais compatíveis. Certifique-se também de que o encaminhamento de IP esteja habilitado para seus nós de processamento. Para obter mais informações, consulte Migrar de `dockershim` para `containerd`.
- Se você já tiver Fluentd configurado para Container Insights, deverá migrar Fluentd para Fluent Bit antes de atualizar seu cluster. Os analisadores Fluentd são configurados para analisar apenas mensagens de log no formato JSON. Ao contrário de `dockerd`, o runtime do `containerd` contêiner tem mensagens de log que não estão no formato JSON. Se você não migrar para Fluent Bit, alguns dos Fluentd's analisadores configurados gerarão muitos erros dentro do Fluentd contêiner. Para obter mais informações sobre migração, consulte Configurar Fluent Bit como um DaemonSet para enviar logs para o CloudWatch Logs.

- No Kubernetes `1.23` e anteriores, certificados servidos pelo `kubelet` com Subject Alternative Names (SANs) IP e DNS não verificáveis são emitidos automaticamente com SANs não verificáveis. Esses SANs não verificáveis são omitidos do certificado provisionado. Em clusters versão `1.24` e posteriores, os certificados servidor pelo `kubelet` não serão emitidos se o SAN não puder ser verificado. Isso impede o funcionamento dos comandos `kubectl` exec e `kubectl` logs. Para obter mais informações, consulte Considerações sobre a assinatura de certificado antes de atualizar seu cluster para o Kubernetes 1.24.
- Ao atualizar um cluster do Amazon EKS `1.23` que usa Fluent Bit, verifique se ele está executando o `k8s/1.3.12` ou posterior. Para isso, reaplique o arquivo YAML Fluent Bit aplicável mais recente peloGitHub. Para obter mais informações, consulte Configurar o Fluent Bit, no Guia do usuário do Amazon CloudWatch.

- Você pode usar dicas sensíveis à topologia para indicar sua preferência por manter o tráfego na zona quando os nós de processamento do cluster são implantados em várias zonas de disponibilidade. O roteamento do tráfego dentro de uma zona pode ajudar a reduzir custos e melhorar a performance da rede. Por padrão, as dicas sensíveis à topologia são habilitadas no Amazon EKS`1.24`. Para obter mais informações, consulte Topology Aware Hints (Dicas sensíveis à topologia) na documentação do Kubernetes.
- O `PodSecurityPolicy` (PSP) está programado para remoção no Kubernetes `1.25`. Os PSPs estão sendo substituídos pelo Pod Security Admission (PSA). O PSA é um controlador de admissão integrado que usa os controles de segurança descritos nos Pod Security Standards (PSS). Tanto o PSA quanto os PSS são recursos beta e são habilitados no Amazon EKS por padrão. Para resolver a remoção da PSP na versão `1.25`, recomendamos que você implemente PSS no Amazon EKS. Para obter mais informações, consulte Implementing Pod Security Standards in Amazon EKS (Implementando Pod Security Standards no Amazon EKS) no blog da AWS.
- A ExecCredential `client.authentication.k8s.io/v1alpha1` foi removida no Kubernetes `1.24`. A API ExecCredential estava disponível em geral no Kubernetes `1.22`. Se você usar um plug-in de credencial client-go que dependa da API `v1alpha1`, entre em contato com o distribuidor do plug-in para saber como migrar para a API `v1`.
- Para o Kubernetes `1.24`, contribuímos com um recurso para o projeto upstream do Cluster Autoscaler que simplifica a escalação de grupos de nós gerenciados do Amazon EKS de e para zero nós. Anteriormente, para que o Cluster Autoscaler entendesse os recursos, rótulos e

taints de um grupo de nós gerenciados que era escalado para zero nós, era necessário marcar o grupo subjacente do Amazon EC2 Auto Scaling com os detalhes dos nós pelos quais ele era responsável. Agora, quando não há nós em execução no grupo de nós gerenciados, o Cluster Autoscaler chama a operação de API `DescribeNodegroup` do Amazon EKS. Essa operação de API fornece as informações de que o Cluster Autoscaler precisa sobre os recursos, rótulos e taints do grupo de nós gerenciados. Esse recurso requer que você adicione a permissão `eks:DescribeNodegroup` à política do IAM da conta do serviço Cluster Autoscaler. Quando o valor de uma tag do Cluster Autoscaler no grupo do Auto Scaling que aciona um grupo de nós gerenciados pelo Amazon EKS conflita com o próprio grupo de nós, o Cluster Autoscaler dá preferência ao valor da tag do grupo do Auto Scaling. Isso permite que você substitua os valores conforme necessário. Para obter mais informações, consulte Escalar a computação em cluster com o Karpenter e o Cluster Autoscaler.

- Se pretender usar os tipos de instância Inferentia ou Trainium com o Amazon EKS `1.24`, você deverá atualizar para o plug-in de dispositivo AWS Neuron versão 1.9.3.0 ou posterior. Para obter mais informações, consulte a versão do Neuron K8 [1.9.3.0] na documentação do AWS Neuron.
- Por padrão, `Containerd` tem o `IPv6` habilitado para Pods. Ele aplica as configurações do kernel do nó aos namespaces de rede do Pod. Por isso, os contêineres em um Pod se vinculam aos endereços de loopback `IPv4` (`127.0.0.1`) and `IPv6` (`::1`). `IPv6` é o protocolo padrão para comunicação. Antes de atualizar o cluster para a versão `1.24`, recomendamos que você teste os Pods de vários contêineres. Modifique os aplicativos para que eles possam se vincular a todos os endereços IP nas interfaces de loopback. A maioria das bibliotecas permite a vinculação do `IPv6`, que é compatível com o `IPv4`. Quando não é possível modificar o código do aplicativo, você tem duas opções:
  - Execute um contêiner `init` e defina `disable ipv6` como `true` (`sysctl -w net.ipv6.conf.all.disable_ipv6=1`).
  - Configure um webhook de admissão mutante para injetar um contêiner `init` junto com os Pods de aplicações.

  Se você precisar bloquear o `IPv6` para todos os Pods em todos os nós, talvez seja necessário desabilitar o `IPv6` nas suas instâncias.
- Por meio do fechamento aleatório de uma conexão, a opção `goaway-chance` no servidor de API do Kubernetes ajuda a evitar que as conexões do cliente `HTTP/2` fiquem presas em uma única instância do servidor de API. Quando a conexão for fechada, o cliente tentará se reconectar e provavelmente acessará um servidor de API diferente como resultado do balanceamento de carga. O Amazon EKS versão `1.24` tem a sinalização `goaway-chance` habilitada. Se sua workload em execução no cluster do Amazon EKS usa um cliente que não é compatível com `HTTP GOAWAY`,

recomendamos atualizar seu cliente para lidar com `GOAWAY` reconectando-se no encerramento da conexão.

Para ver o log de alterações completo do Kubernetes `1.24`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.24.md#changelog-since-v1230.

Kubernetes 1.23

O Kubernetes `1.23` já está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.23`, consulte o anúncio oficial de lançamento.

Important

- O recurso de migração de volume in-tree para a interface de armazenamento de contêiner (CSI) do Kubernetes é habilitado. O recurso permite a substituição de plug-ins de armazenamento em árvore no Kubernetes para o Amazon EBS com um driver de CSI correspondente do Amazon EBS. Para obter mais informações, consulte Recurso do Kubernetes 1.17: migração de volume Kubernetes em árvore para CSI segue para etapa beta no blog Kubernetes.

  O recurso converte APIs em árvore para APIs CSI equivalentes e delega operações a um driver CSI substituto. Com esse recurso, se você usar objetos `StorageClass`, `PersistentVolume` e `PersistentVolumeClaim` que pertençam a essas workloads, provavelmente não haverá nenhuma alteração perceptível. O recurso permite que o Kubernetes delegue todas as operações de gerenciamento de armazenamento do plug-in em árvore para o driver de CSI. Se você usar volumes do Amazon EBS em um cluster existente, instale o driver da CSI do Amazon EBS no cluster antes de atualizá-lo para a versão `1.23`. Se você não instalar o driver antes de atualizar o cluster existente, há a possibilidade de ocorrer interrupções nas workloads. Se você planejar implantar workloads que usem volumes do Amazon EBS em um novo cluster `1.23`, instale o driver da CSI do Amazon EBS no cluster antes de implantar as workloads. Para obter instruções sobre como instalar o driver de CSI do Amazon EBS no cluster, consulte Armazene volumes do Kubernetes com o Amazon EBS. Para ver as perguntas frequentes sobre o recurso de migração, consulte Perguntas frequentes sobre migração de CSI do Amazon EBS.

- O Extended Support para AMIs do Windows otimizadas do Amazon EKS que são publicadas pela AWS não está disponível para o Kubernetes versão`1.23`, mas está disponível para o Kubernetes versão `1.24` e superior.

- O Kubernetes deixou de ser compatível com `dockershim` na versão `1.20` e removeu o `dockershim` na versão `1.24`. Para obter mais informações, consulte O Kubernetes está migrando do Dockershim: compromissos e próximas etapas no blog do Kubernetes. O Amazon EKS deixará de ser compatível com o `dockershim` a partir do Amazon EKS versão `1.24`. A partir do Amazon EKS versão `1.24`, as AMIs oficiais do Amazon EKS terão o `containerd` como o único runtime.

  Embora o Amazon EKS versão `1.23` continue sendo compatível com o `dockershim`, recomendamos que você comece a testar suas aplicações agora mesmo para identificar e remover quaisquer dependências do Docker. Dessa forma, você estará preparado para atualizar seu cluster para a versão `1.24`. Para obter mais informações sobre a remoção do `dockershim`, consulte Migrar de `dockershim` para `containerd`.
- O Kubernetes promoveu para disponibilidade geral a rede de pilha dupla `IPv4`/`IPv6` para Pods, serviços e nós. No entanto, atualmente o Amazon EKS e o Amazon VPC CNI plugin for Kubernetes não são compatíveis com redes de pilha dupla. Seus clusters podem atribuir endereços `IPv4` ou `IPv6` a Pods e serviços, mas não podem atribuir os dois tipos de endereço.
- O Kubernetes promoveu o recurso Pod Security Admission (PSA) para a fase beta. O recurso é habilitado por padrão. Para obter mais informações, consulte Pod Security Admission (Admissão de segurança de pods) na documentação do Kubernetes. O PSA substitui o controlador de admissão Pod Security Policy (PSP). Não há compatibilidade com o controlador de admissão PSP e sua remoção está programada para o Kubernetes versão `1.25`.

  O controlador de admissão PSP aplica padrões de segurança de Pod em Pods em um namespace com base em rótulos específicos de namespace que definem o nível de imposição. Para obter mais informações, consulte Pod Security Standards (PSS) and Pod Security Admission (PSA) (Padrões de segurança de pod [PSS] e admissão de segurança de pod [PSA]) no guia de práticas recomendadas do Amazon EKS.
- Agora, a imagem do `kube-proxy` implantada com clusters é a imagem básica mínima mantida pelo Amazon EKS Distro (EKS-D). A imagem contém pacotes mínimos e não tem shells nem gerenciadores de pacotes.

- O Kubernetes promoveu contêineres efêmeros para a fase beta. Contêineres efêmeros são contêineres temporários executados no mesmo namespace de um Pod existente. Você pode usá-los para observar o estado de Pods e contêineres a fim de solucionar problemas e realizar depuração. Isso é especialmente útil para solucionar problemas interativos quando o `kubectl exec` é insuficiente porque um contêiner sofreu uma pane ou uma imagem de contêiner não inclui utilitários de depuração. As imagens sem distribuição são um exemplo de contêiner que inclui um utilitário de depuração. Para obter mais informações, consulte Debugging with an ephemeral debug container (Depuração com um contêiner efêmero de depuração) na documentação do Kubernetes.
- O Kubernetes promoveu a versão estável da API `HorizontalPodAutoscaler` `autoscaling/v2` para disponibilidade geral. A API `HorizontalPodAutoscaler` `autoscaling/v2beta2` está obsoleta. Ele não está disponível na versão `1.26`.
- Por meio do fechamento aleatório de uma conexão, a opção `goaway-chance` no servidor de API do Kubernetes ajuda a evitar que as conexões do cliente `HTTP/2` fiquem presas em uma única instância do servidor de API. Quando a conexão for fechada, o cliente tentará se reconectar e provavelmente acessará um servidor de API diferente como resultado do balanceamento de carga. O Amazon EKS versão `1.23` tem a sinalização `goaway-chance` habilitada. Se sua workload em execução no cluster do Amazon EKS usa um cliente que não é compatível com `HTTP GOAWAY`, recomendamos atualizar seu cliente para lidar com `GOAWAY` reconectando-se no encerramento da conexão.

Para ver o log de alterações completo do Kubernetes `1.23`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.23.md#changelog-since-v1220.

Revisão das notas de versão do Kubernetes versão 1.22

> **Important**
>
> Não é possível criar novos clusters com essas versões.

Este tópico mostra alterações importantes que você deve conhecer na versão `1.22`. Ao fazer o upgrade, analise cuidadosamente as alterações que ocorreram entre a versão antiga e a nova do seu cluster.

Kubernetes versão **1.22**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.22`: `DefaultStorageClass`, `DefaultTolerationSeconds`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `ResourceQuota`, `ServiceAccount`, `ValidatingAdmissionWebhook`, `PodSecurityPolicy`, `TaintNodesByCondition`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `ExtendedResourceToleration`, `CertificateApproval`, `PodPriority`, `CertificateSigning`, `CertificateSubjectRestriction`, `RuntimeClass` e `DefaultIngressClass`.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
`1.22.17`	`eks.28`	Nova versão da plataforma com correções de segurança e melhorias.	16 de maio de 2024
`1.22.17`	`eks.26`	Nova versão da plataforma com correções de segurança e melhorias.	1º de abril de 2024
`1.22.17`	`eks.14`	Nova versão da plataforma com correções de segurança e melhorias.	30 de junho de 2023
`1.22.17`	`eks.13`	Nova versão da plataforma com correções de segurança e melhorias.	9 de junho de 2023
`1.22.17`	`eks.12`	Nova versão da plataforma com correções de segurança e melhorias.	5 de maio de 2023
`1.22.17`	`eks.11`	Nova versão da plataforma com correções de segurança e melhorias.	24 de março de 2023
`1.22.16`	`eks.10`	Nova versão da plataforma com correções de segurança e melhorias.	27 de janeiro de 2023
`1.22.15`	`eks.9`	Nova versão da plataforma com correções de segurança e melhorias.	5 de dezembro de 2022

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
1.22.15	eks.8	Nova versão da plataforma com correções de segurança e melhorias.	18 de novembro de 2022
1.22.15	eks.7	Nova versão da plataforma com correções de segurança e melhorias.	7 de novembro de 2022
1.22.13	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	21 de setembro de 2022
1.22.10	eks.5	Nova versão da plataforma com melhorias na resiliência do etcd.	15 de agosto de 2022
1.22.10	eks.4	Nova versão da plataforma com correções de segurança e melhorias. Esta versão da plataforma também apresenta um novo controlador de marcação que marca todos os nós de processamento com aws:eks:cluster-name para facilitar a alocação de custos para esses nós de processamento. Para obter mais informações, consulte Marcar recursos para faturamento.	21 de julho de 2022
1.22.10	eks.3	Nova versão da plataforma com correções de segurança e melhorias.	7 de julho de 2022
1.22.9	eks.2	Nova versão da plataforma com correções de segurança e melhorias.	31 de maio de 2022
1.22.6	eks.1	Lançamento inicial do Kubernetes versão 1.22 para o Amazon EKS.	4 de abril de 2022

Visualizar o período atual de suporte do cluster

A seção período de suporte do cluster do console da AWS indica se seu cluster está atualmente em suporte padrão ou estendido. Se o período de suporte do cluster for Suporte estendido, você está sendo cobrado pelo suporte estendido do EKS.

Para obter mais informações sobre o suporte padrão e o suporte estendido, consulte the section called “Versões do Kubernetes”.

Visualizar o período atual de suporte do cluster (Console da AWS)

1. Navegue até a página Clusters na seção EKS do Console da AWS. Confirme se o console está configurado na mesma região da AWS do cluster que deseja revisar.
2. Revise a coluna Período de suporte. Se o valor for Suporte padrão até..., você não está sendo cobrado atualmente pelo suporte estendido. Você está dentro do período de suporte padrão. Se o valor for Suporte estendido..., esse cluster está sendo cobrado pelo suporte estendido.

> **Note**
>
> O Período de suporte não pode ser recuperado com a AWS CLI ou API.

Visualizar a política atual de upgrade do cluster

A política de upgrade de um cluster determina o que acontece quando um cluster chega ao final do período de suporte padrão. Se a política de upgrade for `EXTENDED`, o cluster não será atualizado automaticamente e entrará no suporte estendido. Se política de upgrade for `STANDARD`, ele será atualizado automaticamente.

Os controles do Amazon EKS para a política de versão do Kubernetes permitem escolher o fim do comportamento de suporte padrão para seus clusters do EKS. Com esses controles, você pode decidir quais clusters devem entrar no suporte estendido e quais clusters devem ser atualizados automaticamente ao final do suporte padrão para uma versão do Kubernetes.

Uma versão secundária terá suporte padrão no Amazon EKS durante os primeiros 14 meses após seu lançamento. Depois que uma versão excede a data de término do suporte padrão, ela entra no suporte estendido pelos 12 meses seguintes. O suporte estendido permite que você permaneça em uma Kubernetes versão específica por mais tempo por um custo adicional por hora de cluster. É

possível habilitar ou desabilitar o suporte estendido para um cluster do EKS. Se você desabilitar o suporte estendido, a AWS atualizará automaticamente seu cluster para a próxima versão ao final do suporte padrão. Se você habilitar o suporte estendido, poderá permanecer na versão atual por um custo adicional por um período limitado de tempo. Planeje fazer o upgrade do cluster do Kubernetes regularmente, mesmo se você usa o suporte estendido.

Você pode definir a política de versão para clusters novos e existentes usando a propriedade `supportType`. Há duas opções que podem ser usadas para definir a política de suporte a versões:

- **`STANDARD`**: o cluster do EKS está qualificado para upgrade automático ao final do suporte padrão. Você não incorrerá em cobranças de suporte estendido com essa configuração, mas seu cluster do EKS será atualizado automaticamente para a próxima versão do Kubernetes compatível com o suporte padrão.
- **`EXTENDED`**: o cluster do EKS entrará em suporte estendido quando a versão do Kubernetes atingir o fim do suporte padrão. Você incorrerá em taxas de suporte estendidas com essa configuração. É possível atualizar o cluster para uma versão do Kubernetes compatível com o suporte padrão parar de incorrer em cobranças de suporte estendido. Clusters executados com suporte estendido estarão qualificados para atualização automática ao final do suporte estendido.

O suporte estendido está habilitado por padrão para todos os novos clusters e para os clusters existentes. É possível verificar se o suporte estendido está habilitado para um cluster via AWS Management Console ou AWS CLI.

> **Important**
>
> Se você quiser que seu cluster permaneça na versão Kubernetes atual para se beneficiar do período de suporte estendido, será necessário habilitar a política de atualização de suporte estendido antes do final do período de suporte padrão.

Só é possível definir a política de suporte de versão para seus clusters enquanto ela é executada na versão do Kubernetes em suporte padrão. Depois que a versão entrar no suporte estendido, você não poderá alterar essa configuração até que execute uma versão no suporte padrão.

Por exemplo, se você definiu sua política de suporte de versão como `standard`, não poderá alterar essa configuração depois que a versão do Kubernetes em execução no seu cluster atingir o fim do suporte padrão. Se você definiu sua política de suporte de versão como `extended`, não poderá alterar essa configuração depois que a versão do Kubernetes em execução no seu cluster atingir o

fim do suporte padrão. Para alterar a configuração da política de suporte de versão, seu cluster deve estar sendo executado em uma versão padrão compatível do Kubernetes.

Visualizar a política de upgrade do cluster (Console da AWS)

1. Navegue até a página Clusters na seção EKS do Console da AWS. Confirme se o console está configurado na mesma região da AWS do cluster que deseja revisar.
2. Revise a coluna Política de upgrade. Se o valor for Suporte padrão, o cluster não entrará no suporte estendido. Se o valor for Suporte estendido, o cluster entrará no suporte estendido.

Visualizar a política de upgrade do cluster (AWS CLI)

1. Verifique se a AWS CLI está instalada e se você está conectado. Saiba como atualizar e instalar a AWS CLI.
2. Determine o nome do seu cluster do EKS. Defina a CLI para a mesma região da AWS do seu cluster do EKS.
3. Execute o seguinte comando:

```
aws eks describe-cluster \
--name <cluster-name> \
--query "cluster.upgradePolicy.supportType"
```

4. Se o valor for `STANDARD`, o cluster não entrará no suporte estendido. Se o valor for `EXTENDED`, o cluster entrará no suporte estendido.

Adicione flexibilidade para planejar atualizações de versão do Kubernetes habilitando o suporte estendido do EKS

Este tópico descreve como definir a política de atualização de um cluster do EKS para habilitar o suporte estendido. A política de atualização de um cluster do EKS determina o que acontece quando um cluster chega ao final do período de suporte padrão. Se uma política de upgrade de cluster tiver o suporte estendido habilitado, ela entrará no período de suporte estendido no final do período de suporte padrão. O cluster não será atualizado automaticamente no final do período de suporte padrão.

Na verdade, os clusters, no período de suporte estendido, incorrem em custos mais altos. Se um cluster simplesmente tiver a política de atualização definida para permitir o suporte estendido e estiver no período de suporte padrão, ele incorrerá em custos padrão.

Os clusters do EKS têm a política de atualização definida para habilitar o suporte estendido por padrão.

Para obter mais informações sobre políticas de atualização, consulte the section called “Visualizar política de upgrade”.

> **Important**
>
> Se você quiser que seu cluster permaneça na versão Kubernetes atual para se beneficiar do período de suporte estendido, será necessário habilitar a política de atualização de suporte estendido antes do final do período de suporte padrão.
> Se você não ativar o suporte estendido, seu cluster será atualizado automaticamente.

Habilitar suporte estendido do EKS (Console da AWS)

1. Navegue para seu cluster do EKS no Console da AWS. Selecione a guia Visão geral na página Informações do cluster.
2. Na seção Configurações de versão do Kubernetes, selecione Gerenciar.
3. Selecione Suporte estendido e, em seguida, Salvar alterações.

Habilitar suporte estendido do EKS (AWS CLI)

1. Verifique se a AWS CLI está instalada e se você está conectado. Saiba como atualizar e instalar a AWS CLI.
2. Determine o nome do seu cluster do EKS.
3. Execute o seguinte comando:

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType = EXTENDED
```

Evitar o aumento dos custos do cluster desabilitando o suporte estendido do EKS

Este tópico descreve como definir a política de atualização de um cluster do EKS para desabilitar o suporte estendido. A política de atualização de um cluster do EKS determina o que acontece quando um cluster chega ao final do período de suporte padrão. Se uma política de atualização de cluster tiver o suporte estendido desabilitado, ela será atualizada automaticamente para a próxima versão do Kubernetes.

Para obter mais informações sobre políticas de atualização, consulte the section called “Visualizar política de upgrade”.

> **Important**
>
> Não é possível desabilitar o suporte estendido após o cluster entrar nele. O suporte estendido pode ser desabilitado somente para clusters no suporte padrão.
> A AWS recomenda atualizar o cluster para uma versão no período de suporte padrão.

Desabilitar suporte estendido do EKS (Console da AWS)

1. Navegue para seu cluster do EKS no Console da AWS. Selecione a guia Visão geral na página Informações do cluster.
2. Na seção Configurações de versão do Kubernetes, selecione Gerenciar.
3. Selecione Suporte padrão e, em seguida, Salvar alterações.

Desabilitar suporte estendido do EKS (AWS CLI)

1. Verifique se a AWS CLI está instalada e se você está conectado. Saiba como atualizar e instalar a AWS CLI.
2. Determine o nome do seu cluster do EKS.
3. Execute o seguinte comando:

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType = STANDARD
```

Veja as versões da plataforma do Amazon EKS para cada versão do Kubernetes

As versões da plataforma Amazon EKS representam os recursos do ambiente de gerenciamento do cluster do Amazon EKS, como quais sinalizadores do servidor de API do Kubernetes estão habilitados e também a versão de patch atual do Kubernetes. Toda versão secundária do Kubernetes tem uma ou mais versões de plataforma do Amazon EKS associadas. As versões de plataforma para diferentes versões secundárias do Kubernetes são independentes. Você pode recuperar a versão atual da plataforma do seu cluster usando a AWS CLI ou o AWS Management Console. Se você tiver um cluster local no AWS Outposts, consulte Conheça as versões de plataforma do Kubernetes e do Amazon EKS para AWS Outposts em vez deste tópico.

Quando uma nova versão secundária do Kubernetes estiver disponível no Amazon EKS, como a 1.30, a versão inicial da plataforma do Amazon EKS para essa versão secundária do Kubernetes começará em `eks.1`. Porém, o Amazon EKS lança novas versões de plataforma periodicamente para habilitar novas configurações do ambiente de gerenciamento do Kubernetes e para fornecer correções de segurança.

Quando novas versões de plataforma do Amazon EKS são disponibilizadas para uma versão secundária:

- O número da versão da plataforma do Amazon EKS é incrementado (`eks.`*`n+1`*).
- O Amazon EKS atualiza automaticamente todos os clusters existentes da versão de plataforma mais recente do Amazon EKS para a versão secundária correspondente do Kubernetes. As atualizações automáticas de versões de plataformas existentes do Amazon EKS são implementadas de forma incremental. O processo de implantação pode levar algum tempo. Se precisar dos recursos da versão de plataforma mais recente do Amazon EKS imediatamente, você deverá criar outro cluster do Amazon EKS.

  Se o cluster tiver mais de duas versões de plataforma atrás da versão atual da plataforma, é possível que o Amazon EKS não tenha conseguido atualizar automaticamente o cluster. Para obter detalhes sobre o que pode causar isso, consulte A versão da plataforma Amazon EKS está mais de duas versões atrás da versão atual da plataforma.
- O Amazon EKS pode publicar uma nova AMI do nó com uma versão de patch correspondente. Porém, todas as versões de patch são compatíveis entre o ambiente de gerenciamento do EKS e as AMIs do nó para uma determinada versão secundária do Kubernetes.

Novas versões de plataforma do Amazon EKS não apresentam alterações que podem causar interrupções nem causam interrupções do serviço.

Os clusters são sempre criados com a versão de plataforma mais recente disponível do Amazon EKS (`eks.n`) para a versão especificada do Kubernetes. Se você atualizar o cluster para uma nova versão secundária do Kubernetes, o cluster receberá a versão atual da plataforma do Amazon EKS para a versão secundária do Kubernetes para a qual você atualizou.

As versões recentes e atuais da plataforma do Amazon EKS estão descritas nas tabelas a seguir.

> **Note**
>
> A AWS desabilitou recentemente algumas versões da plataforma publicadas em junho de 2024. As versões da plataforma apresentavam problemas de estabilidade. Nenhuma ação é necessária.

Kubernetes versão **`1.30`**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.30`: `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
`1.30.2`	`eks.5`	Nova versão da plataforma com correções de segurança e melhorias.	2 de julho de 2024
`1.30.0`	`eks.2`	Lançamento inicial do Kubernetes versão `1.30` para o EKS. Para obter mais informações, consulte Kubernetes 1.30.	23 de maio de 2024

Kubernetes versão **1.29**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.29`: `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
`1.29.6`	`eks.10`	Nova versão da plataforma com correções de segurança e melhorias.	2 de julho de 2024
`1.29.4`	`eks.7`	Nova versão da plataforma com autoescalabilidade do CoreDNS, correções de segurança e melhorias . Para obter mais informações sobre autoescalabilidade do CordeDNS, consulte Autoescalabilidade do CoreDNS.	16 de maio de 2024
`1.29.3`	`eks.6`	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2024
`1.29.1`	`eks.5`	Nova versão da plataforma com correções de segurança e melhorias.	29 de março de 2024
`1.29.1`	`eks.4`	Nova versão da plataforma com correções de segurança e melhorias.	20 de março de 2024
`1.29.1`	`eks.3`	Nova versão da plataforma com correções de segurança e melhorias.	12 de março de 2024
`1.29.0`	`eks.1`	Lançamento inicial do Kubernete s versão `1.29` para o EKS. Para	23 de janeiro de 2024

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
		obter mais informações, consulte Kubernetes 1.29.	

Kubernetes versão **1.28**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.28`: `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
`1.28.11`	`eks.16`	Nova versão da plataforma com correções de segurança e melhorias.	2 de julho de 2024
`1.28.9`	`eks.13`	Nova versão da plataforma com autoescalabilidade do CoreDNS, correções de segurança e melhorias . Para obter mais informações sobre autoescalabilidade do CordeDNS, consulte Autoescalabilidade do CoreDNS.	16 de maio de 2024
`1.28.8`	`eks.12`	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2024
`1.28.7`	`eks.11`	Nova versão da plataforma com correções de segurança e melhorias.	29 de março de 2024

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.28.7	eks.10	Nova versão da plataforma com correções de segurança e melhorias.	20 de março de 2024
1.28.6	eks.9	Nova versão da plataforma com correções de segurança e melhorias.	12 de março de 2024
1.28.5	eks.7	Nova versão da plataforma com correções de segurança e melhorias.	17 de janeiro de 2024
1.28.4	eks.6	Nova versão da plataforma com entradas de acesso, correções de segurança e melhorias.	14 de dezembro de 2023
1.28.4	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	12 de dezembro de 2023
1.28.3	eks.4	Nova versão da plataforma com Saiba como a EKS Pod Identity concede aos pods acesso aos serviços da AWS, correções de segurança e melhorias.	10 de novembro de 2023
1.28.3	eks.3	Nova versão da plataforma com correções de segurança e melhorias.	3 de novembro de 2023
1.28.2	eks.2	Nova versão da plataforma com correções de segurança e melhorias.	16 de outubro de 2023
1.28.1	eks.1	Lançamento inicial do Kubernetes versão 1.28 para o EKS. Para obter mais informações, consulte Kubernetes 1.28.	26 de setembro de 2023

Kubernetes versão **1.27**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.27`: `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
`1.27.15`	`eks.20`	Nova versão da plataforma com correções de segurança e melhorias.	2 de julho de 2024
`1.27.13`	`eks.17`	Nova versão da plataforma com autoescalabilidade do CoreDNS, correções de segurança e melhorias . Para obter mais informações sobre autoescalabilidade do CordeDNS, consulte Autoescalabilidade do CoreDNS.	16 de maio de 2024
`1.27.12`	`eks.16`	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2024
`1.27.11`	`eks.15`	Nova versão da plataforma com correções de segurança e melhorias.	29 de março de 2024
`1.27.11`	`eks.14`	Nova versão da plataforma com correções de segurança e melhorias.	20 de março de 2024
`1.27.10`	`eks.13`	Nova versão da plataforma com correções de segurança e melhorias.	12 de março de 2024
`1.27.9`	`eks.11`	Nova versão da plataforma com correções de segurança e melhorias.	17 de janeiro de 2024

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.27.8	eks.10	Nova versão da plataforma com entradas de acesso, correções de segurança e melhorias.	14 de dezembro de 2023
1.27.8	eks.9	Nova versão da plataforma com correções de segurança e melhorias.	12 de dezembro de 2023
1.27.7	eks.8	Nova versão da plataforma com Saiba como a EKS Pod Identity concede aos pods acesso aos serviços da AWS, correções de segurança e melhorias.	10 de novembro de 2023
1.27.7	eks.7	Nova versão da plataforma com correções de segurança e melhorias.	3 de novembro de 2023
1.27.6	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	16 de outubro de 2023
1.27.4	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	30 de agosto de 2023
1.27.4	eks.4	Nova versão da plataforma com correções de segurança e melhorias.	30 de julho de 2023
1.27.3	eks.3	Nova versão da plataforma com correções de segurança e melhorias.	30 de junho de 2023
1.27.2	eks.2	Nova versão da plataforma com correções de segurança e melhorias.	9 de junho de 2023
1.27.1	eks.1	Lançamento inicial do Kubernetes versão 1.27 para o EKS. Para obter mais informações, consulte Kubernetes 1.27.	24 de maio de 2023

Kubernetes versão **1.26**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.26`: `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
`1.26.15`	`eks.21`	Nova versão da plataforma com correções de segurança e melhorias.	2 de julho de 2024
`1.26.15`	`eks.18`	Nova versão da plataforma com autoescalabilidade do CoreDNS, correções de segurança e melhorias . Para obter mais informações sobre autoescalabilidade do CordeDNS, consulte Autoescalabilidade do CoreDNS.	16 de maio de 2024
`1.26.15`	`eks.17`	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2024
`1.26.14`	`eks.16`	Nova versão da plataforma com correções de segurança e melhorias.	29 de março de 2024
`1.26.14`	`eks.15`	Nova versão da plataforma com correções de segurança e melhorias.	20 de março de 2024
`1.26.13`	`eks.14`	Nova versão da plataforma com correções de segurança e melhorias.	12 de março de 2024
`1.26.12`	`eks.12`	Nova versão da plataforma com correções de segurança e melhorias.	17 de janeiro de 2024

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.26.11	eks.11	Nova versão da plataforma com entradas de acesso, correções de segurança e melhorias.	14 de dezembro de 2023
1.26.11	eks.10	Nova versão da plataforma com correções de segurança e melhorias.	12 de dezembro de 2023
1.26.10	eks.9	Nova versão da plataforma com Saiba como a EKS Pod Identity concede aos pods acesso aos serviços da AWS, correções de segurança e melhorias.	10 de novembro de 2023
1.26.10	eks.8	Nova versão da plataforma com correções de segurança e melhorias.	3 de novembro de 2023
1.26.9	eks.7	Nova versão da plataforma com correções de segurança e melhorias.	16 de outubro de 2023
1.26.7	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	30 de agosto de 2023
1.26.7	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	30 de julho de 2023
1.26.6	eks.4	Nova versão da plataforma com correções de segurança e melhorias.	30 de junho de 2023
1.26.5	eks.3	Nova versão da plataforma com correções de segurança e melhorias.	9 de junho de 2023
1.26.4	eks.2	Nova versão da plataforma com correções de segurança e melhorias.	5 de maio de 2023

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.26.2	eks.1	Lançamento inicial do Kubernetes versão 1.26 para o EKS. Para obter mais informações, consulte Kubernetes 1.26.	11 de abril de 2023

Kubernetes versão **1.25**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.25: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota.

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.25.16	eks.22	Nova versão da plataforma com correções de segurança e melhorias.	2 de julho de 2024
1.25.16	eks.19	Nova versão da plataforma com autoescalabilidade do CoreDNS, correções de segurança e melhorias. Para obter mais informações sobre autoescalabilidade do CordeDNS, consulte Autoescalabilidade do CoreDNS.	16 de maio de 2024
1.25.16	eks.18	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2024
1.25.16	eks.17	Nova versão da plataforma com correções de segurança e melhorias.	29 de março de 2024

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.25.16	eks.16	Nova versão da plataforma com correções de segurança e melhorias.	20 de março de 2024
1.25.16	eks.15	Nova versão da plataforma com correções de segurança e melhorias.	12 de março de 2024
1.25.16	eks.13	Nova versão da plataforma com correções de segurança e melhorias.	17 de janeiro de 2024
1.25.16	eks.12	Nova versão da plataforma com entradas de acesso, correções de segurança e melhorias.	14 de dezembro de 2023
1.25.16	eks.11	Nova versão da plataforma com correções de segurança e melhorias.	12 de dezembro de 2023
1.25.15	eks.10	Nova versão da plataforma com Saiba como a EKS Pod Identity concede aos pods acesso aos serviços da AWS, correções de segurança e melhorias.	10 de novembro de 2023
1.25.15	eks.9	Nova versão da plataforma com correções de segurança e melhorias.	3 de novembro de 2023
1.25.14	eks.8	Nova versão da plataforma com correções de segurança e melhorias.	16 de outubro de 2023
1.25.12	eks.7	Nova versão da plataforma com correções de segurança e melhorias.	30 de agosto de 2023
1.25.12	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	30 de julho de 2023
1.25.11	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	30 de junho de 2023

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.25.10	eks.4	Nova versão da plataforma com correções de segurança e melhorias.	9 de junho de 2023
1.25.9	eks.3	Nova versão da plataforma com correções de segurança e melhorias.	5 de maio de 2023
1.25.8	eks.2	Nova versão da plataforma com correções de segurança e melhorias.	24 de março de 2023
1.25.6	eks.1	Lançamento inicial do Kubernetes versão 1.25 para o EKS. Para obter mais informações, consulte Kubernetes 1.25.	21 de fevereiro de 2023

Kubernetes versão **1.24**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.24: CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, Priority, PodSecurityPolicy, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition e ValidatingAdmissionWebhook.

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.24.17	eks.25	Nova versão da plataforma com correções de segurança e melhorias.	2 de julho de 2024
1.24.17	eks.22	Nova versão da plataforma com correções de segurança e melhorias.	16 de maio de 2024

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.24.17	eks.21	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2024
1.24.17	eks.20	Nova versão da plataforma com correções de segurança e melhorias.	29 de março de 2024
1.24.17	eks.19	Nova versão da plataforma com correções de segurança e melhorias.	20 de março de 2024
1.24.17	eks.18	Nova versão da plataforma com correções de segurança e melhorias.	12 de março de 2024
1.24.17	eks.16	Nova versão da plataforma com correções de segurança e melhorias.	17 de janeiro de 2024
1.24.17	eks.15	Nova versão da plataforma com entradas de acesso, correções de segurança e melhorias.	14 de dezembro de 2023
1.24.17	eks.14	Nova versão da plataforma com correções de segurança e melhorias.	12 de dezembro de 2023
1.24.17	eks.13	Nova versão da plataforma com Saiba como a EKS Pod Identity concede aos pods acesso aos serviços da AWS, correções de segurança e melhorias.	10 de novembro de 2023
1.24.17	eks.12	Nova versão da plataforma com correções de segurança e melhorias.	3 de novembro de 2023
1.24.17	eks.11	Nova versão da plataforma com correções de segurança e melhorias.	16 de outubro de 2023
1.24.16	eks.10	Nova versão da plataforma com correções de segurança e melhorias.	30 de agosto de 2023

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.24.16	eks.9	Nova versão da plataforma com correções de segurança e melhorias.	30 de julho de 2023
1.24.15	eks.8	Nova versão da plataforma com correções de segurança e melhorias.	30 de junho de 2023
1.24.14	eks.7	Nova versão da plataforma com correções de segurança e melhorias.	9 de junho de 2023
1.24.13	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	5 de maio de 2023
1.24.12	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	24 de março de 2023
1.24.8	eks.4	Nova versão da plataforma com correções de segurança e melhorias.	27 de janeiro de 2023
1.24.7	eks.3	Nova versão da plataforma com correções de segurança e melhorias.	5 de dezembro de 2022
1.24.7	eks.2	Nova versão da plataforma com correções de segurança e melhorias.	18 de novembro de 2022
1.24.7	eks.1	Lançamento inicial do Kubernetes versão 1.24 para o EKS. Para obter mais informações, consulte Kubernetes 1.24.	15 de novembro de 2022

Kubernetes versão **1.23**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.23: CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook,

`NamespaceLifecycle`, `NodeRestriction`, `PersistentVolumeClaimResize`, `Priority`, `PodSecurityPolicy`, `ResourceQuota`, `RuntimeClass`, `ServiceAccount`, `StorageObjectInUseProtection`, `TaintNodesByCondition` e `ValidatingAdmissionWebhook`.

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
`1.23.17`	`eks.27`	Nova versão da plataforma com correções de segurança e melhorias.	2 de julho de 2024
`1.23.17`	`eks.24`	Nova versão da plataforma com correções de segurança e melhorias.	16 de maio de 2024
`1.23.17`	`eks.23`	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2024
`1.23.17`	`eks.22`	Nova versão da plataforma com correções de segurança e melhorias.	29 de março de 2024
`1.23.17`	`eks.21`	Nova versão da plataforma com correções de segurança e melhorias.	20 de março de 2024
`1.23.17`	`eks.20`	Nova versão da plataforma com correções de segurança e melhorias.	12 de março de 2024
`1.23.17`	`eks.18`	Nova versão da plataforma com correções de segurança e melhorias.	17 de janeiro de 2024
`1.23.17`	`eks.17`	Nova versão da plataforma com entradas de acesso, correções de segurança e melhorias.	14 de dezembro de 2023
`1.23.17`	`eks.16`	Nova versão da plataforma com correções de segurança e melhorias.	12 de dezembro de 2023
`1.23.17`	`eks.15`	Nova versão da plataforma com correções de segurança e melhorias.	10 de novembro de 2023

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
1.23.17	eks.14	Nova versão da plataforma com correções de segurança e melhorias.	3 de novembro de 2023
1.23.17	eks.13	Nova versão da plataforma com correções de segurança e melhorias.	16 de outubro de 2023
1.23.17	eks.12	Nova versão da plataforma com correções de segurança e melhorias.	30 de agosto de 2023
1.23.17	eks.11	Nova versão da plataforma com correções de segurança e melhorias.	30 de julho de 2023
1.23.17	eks.10	Nova versão da plataforma com correções de segurança e melhorias.	30 de junho de 2023
1.23.17	eks.9	Nova versão da plataforma com correções de segurança e melhorias.	9 de junho de 2023
1.23.17	eks.8	Nova versão da plataforma com correções de segurança e melhorias.	5 de maio de 2023
1.23.17	eks.7	Nova versão da plataforma com correções de segurança e melhorias.	24 de março de 2023
1.23.14	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	27 de janeiro de 2023
1.23.13	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	5 de dezembro de 2022
1.23.13	eks.4	Nova versão da plataforma com correções de segurança e melhorias.	18 de novembro de 2022
1.23.12	eks.3	Nova versão da plataforma com correções de segurança e melhorias.	7 de novembro de 2022

Versão do Kubernetes	Versão da plataforma EKS	Notas de release	Data de lançamento
`1.23.10`	`eks.2`	Nova versão da plataforma com correções de segurança e melhorias.	21 de setembro de 2022
`1.23.7`	`eks.1`	Lançamento inicial do Kubernetes versão `1.23` para o EKS. Para obter mais informações, consulte Kubernetes 1.23.	11 de agosto de 2022

Obter a versão atual da plataforma

Para obter a versão atual da plataforma para seu cluster (console)

1. Abra o console do Amazon EKS.
2. No painel de navegação, escolha Clusters.
3. Na lista de clusters, escolha o Nome do cluster para verificar a versão da plataforma.
4. Escolha a guia Overview (Visão geral).
5. A Versão da plataforma está disponível na seção Detalhes.

Para obter a versão atual da plataforma para seu cluster (AWS CLI)

1. Determine o Nome do cluster do qual você deseja verificar a versão da plataforma.
2. Execute o seguinte comando:

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

Veja um exemplo de saída abaixo.

```
"eks.10"
```

Alterar versão da plataforma

Não é possível alterar a versão da plataforma de um cluster do EKS. Quando novas versões de plataforma do Amazon EKS são disponibilizadas para uma versão do Kubernetes, o EKS atualiza automaticamente todos os clusters existentes para a versão de plataforma mais recente do Amazon EKS para a versão do Kubernetes correspondente. As atualizações automáticas de versões de plataformas existentes do Amazon EKS são implementadas de forma incremental. Você não pode usar o Console da AWS ou a CLI para alterar a versão da plataforma.

Se você atualizar a versão do Kubernetes, seu cluster migrará para a versão mais recente da plataforma para a versão do Kubernetes.

Escalar a computação em cluster com o Karpenter e o Cluster Autoscaler

A escalação automática é uma função que aumenta ou reduz automaticamente a escala dos recursos para atender a alterações de demanda. É uma função importante do Kubernetes cuja execução manual exigiria extensos recursos humanos.

O Amazon EKS oferece suporte a dois produtos de escalabilidade automática:

Karpenter

> O Karpenter é um autoescalador de cluster do Kubernetes, flexível e de alto desempenho, que ajuda a melhorar a disponibilidade das aplicações e a eficiência do cluster. O Karpenter inicia a quantidade certa de recursos computacionais (por exemplo, instâncias do Amazon EC2) em resposta a alterações na carga da aplicação em menos de um minuto. Por meio da integração do Kubernetes com o AWS, o Karpenter pode provisionar recursos computacionais just-in-time que atendam perfeitamente aos requisitos da workload. O Karpenter provisiona automaticamente novos recursos computacionais com base nos requisitos específicos das workloads do cluster. Isso inclui requisitos de computação, armazenamento, aceleração e agendamento. O Amazon EKS é compatível com clusters que usam o Karpenter, embora o Karpenterfuncione com qualquer cluster compatível do Kubernetes. Para obter mais informações, consulte a documentação do Karpenter.

Autoscaler do cluster

O autoescalador de cluster do Kubernetes ajusta automaticamente o número de nós do cluster quando os pods falham ou são reagendados para outros nós. O Cluster Autoscaler usa grupos do Auto Scaling. Para obter mais informações, consulte Autoscaler do cluster na AWS.

Saiba como o controle de acesso funciona no Amazon EKS

Saiba como gerenciar o acesso ao seu cluster do Amazon EKS. O uso do Amazon EKS requer conhecimento de como o Kubernetes e o AWS Identity and Access Management (AWS IAM) lidam com o controle de acesso.

Esta seção inclui:

the section called “Conceder acesso a APIs do Kubernetes”: saiba como permitir que aplicações ou usuários se autentiquem na API do Kubernetes. É possível usar entradas de acesso, o aws-auth ConfigMap ou um provedor de OIDC externo.

the section called “Acessar meu cluster com o kubectl”: saiba como configurar o kubectl para se comunicar com o seu cluster do Amazon EKS. Use a AWS CLI para criar um arquivo kubeconfig.

the section called “Conceder às workloads acesso à AWS”: saiba como associar uma conta de serviço do Kubernetes a perfis do AWS IAM. Você pode usar a Identidade de Pods ou perfis do IAM para contas de serviço (IRSA).

Tarefas comuns:

- Conceda aos desenvolvedores acesso à API do Kubernetes. Visualize recursos do Kubernetes no AWS Management Console.
  - Solução: use entradas de acesso para associar permissões de Kubernetes RBAC a usuários ou perfis do AWS IAM.
- Configure o kubectl para se comunicar com um cluster do Amazon EKS usando credenciais da AWS.
  - Solução: usar a AWS CLI para criar um arquivo kubeconfig.
- Use um provedor de identidade externo, como o Ping Identity, para autenticar usuários na API do Kubernetes.
  - Solução: vincule um provedor de OIDC externo.
- Conceda às workloads em seu cluster do Kubernetes a capacidade de chamar APIs da AWS.
  - Solução: use a Identidade de Pods para associar um perfil do AWS IAM a uma conta de serviço da Kubernetes.

Contexto:

- Saiba como as contas de serviço do Kubernetes funcionam.
- Analise o modelo de controle de acesso baseado em perfil (RBAC) do Kubernetes
- Para obter mais informações sobre como gerenciar o acesso a recursos da AWS, consulte o Guia do usuário do AWS IAM. Como alternativa, faça um treinamento introdutório gratuito sobre o uso do AWS IAM.

Conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes

Seu cluster tem um endpoint de API do Kubernetes. O Kubectl usa essa API. Você pode se autenticar nessa API usando dois tipos de identidades:

- Uma entidade principal (IAM) do AWS Identity and Access Management (perfil ou usuário): este tipo requer autenticação do IAM. Os usuários podem fazer login na AWS como um usuário do IAM ou com uma identidade federada ao usar credenciais fornecidas por meio de uma fonte de identidades. Os usuários só poderão fazer login com uma identidade federada se o administrador tiver configurado a federação de identidades anteriormente usando perfis do IAM. Quando os usuários acessam a AWS usando a federação, estão indiretamente assumindo um perfil. Quando os usuários usam esse tipo de identidade, você:
  - Pode atribuir permissões do Kubernetes a eles para que eles possam trabalhar com objetos do Kubernetes no seu cluster. Para obter mais informações sobre como atribuir permissões às entidades principais do IAM para que elas possam acessar objetos do Kubernetes no seu cluster, consulte Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS.
  - Pode atribuir a eles permissões do IAM para que eles possam trabalhar com seu cluster do Amazon EKS e seus recursos usando a API do Amazon EKS, AWS CLI, AWS CloudFormation, AWS Management Console ou `eksctl`. Para obter mais informações, consulte Ações definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização do serviço.
  - Os nós se juntam ao seu cluster assumindo um perfil do IAM. A habilidade de acessar o cluster usando as entidades principais do IAM é fornecida pelo AWS IAM Authenticator para Kubernetes, que é executado no ambiente de gerenciamento do Amazon EKS.
- Um usuário em seu próprio provedor OpenID Connect (OIDC): este tipo requer autenticação em seu provedor OIDC. Para obter mais informações sobre como configurar o seu próprio provedor

OIDC com o seu cluster do Amazon EKS, consulte Conceda aos usuários acesso ao Kubernetes com um provedor OIDC externo. Quando os usuários usam esse tipo de identidade, você:

- Pode atribuir permissões do Kubernetes a eles para que eles possam trabalhar com objetos do Kubernetes no seu cluster.
- Não pode atribuir a eles permissões do IAM para que eles possam trabalhar com seu cluster do Amazon EKS e seus recursos usando a API do Amazon EKS, AWS CLI, AWS CloudFormation, AWS Management Console ou `eksctl`.

Você pode usar os dois tipos de identidades com seu cluster. Não é possível desabilitar o método de autenticação do IAM. O método de autenticação OIDC é opcional.

Associe identidades do IAM a permissões do Kubernetes

O AWS IAM Authenticator para Kubernetes está instalado no ambiente de gerenciamento do cluster. Ele habilita entidades principais (perfis e usuários) do AWS Identity and Access Management (IAM) às quais você permite o acesso a recursos do Kubernetes em seu cluster. É possível permitir que entidades principais do IAM acessem objetos do Kubernetes no seu cluster usando um dos seguintes métodos:

- Criação de entradas de acesso: se seu cluster for igual ou posterior à versão da plataforma listada na seção Pré-requisitos da versão do Kubernetes do cluster, recomendamos que você use essa opção.

  Use entradas de acesso para gerenciar as permissões do Kubernetes das entidades principais do IAM de fora do cluster. É possível adicionar e gerenciar o acesso ao cluster usando a API do EKS, a AWS Command Line Interface, os SDKs da AWS, o AWS CloudFormation e o AWS Management Console. Isso significa que você pode gerenciar usuários com as mesmas ferramentas com as quais criou o cluster.

  Para começar, siga Alterar o modo de autenticação para usar entradas de acesso e, depois, Migrando entradas existentes `aws-auth ConfigMap` para entradas de acesso.
- Adição de entradas ao **`aws-auth ConfigMap`**: se a versão da plataforma do seu cluster for anterior à versão listada na seção Pré-requisitos, você deverá usar esta opção. Se a versão da plataforma do seu cluster for igual ou posterior à versão da plataforma listada na seção Pré-requisitos da versão do Kubernetes do cluster e você tiver adicionado entradas ao `ConfigMap`, recomendamos que migre essas entradas para entradas de acesso. No entanto, você não pode migrar entradas que o Amazon EKS adicionou ao `ConfigMap`, como entradas para perfis do

IAM usadas com grupos de nós gerenciados ou perfis do Fargate. Para obter mais informações, consulte the section called “Conceder acesso a APIs do Kubernetes”.

- Caso precise usar a opção do `aws-auth ConfigMap`, você poderá adicionar entradas ao `ConfigMap` usando o comando **`eksctl create iamidentitymapping`**. Para obter mais informações, consulte Gerenciar usuários e perfis do IAM na documentação do `eksctl`.

Definir modo de autenticação do cluster

Cada cluster tem um modo de autenticação. O modo de autenticação determina quais métodos você pode usar para permitir que as entidades principais do IAM acessem os objetos do Kubernetes no seu cluster. Existem três modos de autenticação.

> **Important**
>
> Depois que o método de entrada de acesso estiver habilitado, ele não poderá ser desabilitado.
> Se o método `ConfigMap` não for habilitado durante a criação do cluster, ele não poderá ser habilitado posteriormente. Todos os clusters criados antes da introdução das entradas de acesso têm o método `ConfigMap` habilitado.

O `aws-auth ConfigMap` dentro do cluster

Este é o modo de autenticação original para clusters do Amazon EKS. A entidade principal do IAM que criou o cluster é o usuário inicial que pode acessar o cluster usando `kubectl`. O usuário inicial deve adicionar outros usuários à lista no `aws-auth ConfigMap` e atribuir permissões que afetem os outros usuários dentro do cluster. Os outros usuários não podem gerenciar nem remover o usuário inicial, pois não há uma entrada no `ConfigMap` para gerenciar.

Tanto as entradas do `ConfigMap` quanto as de acesso

Com este modo de autenticação, é possível usar os dois métodos para adicionar entidades principais do IAM ao cluster. Observe que cada método armazena entradas separadas; por exemplo, se você adicionar uma entrada de acesso da AWS CLI, o `aws-auth ConfigMap` não será atualizado.

Somente entradas de acesso

Com este modo de autenticação, é possível usar a API do EKS, a AWS Command Line Interface, os SDKs da AWS, o AWS CloudFormation e o AWS Management Console para gerenciar o acesso ao cluster para entidades principais do IAM.

Cada entrada de acesso tem um tipo e você pode usar a combinação de um escopo de acesso para limitar a entidade principal a um namespace específico e uma política de acesso para definir políticas de permissões reutilizáveis pré-configuradas. Como alternativa, você pode usar o tipo STANDARD e grupos de RBAC do Kubernetes para atribuir permissões personalizadas.

Modo de autenticação	Métodos
Somente `ConfigMap` (`CONFIG_MAP`)	`aws-auth` `ConfigMap`
API do EKS e `ConfigMap` (`API_AND_CONFIG_MAP`)	entradas de acesso na API do EKS, AWS Command Line Interface, SDKs da AWS, AWS CloudFormation e AWS Management Console e `aws-auth` `ConfigMap`
Somente API do EKS (`API`)	entradas de acesso na API do EKS, AWS Command Line Interface, SDKs da AWS, AWS CloudFormation e AWS Management Console

Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS

Pré-requisitos

- Familiaridade com as opções de acesso ao cluster do Amazon EKS. Para obter mais informações, consulte Conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes.
- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS. Para usar entradas de acesso e mudar o modo de autenticação de um cluster, o cluster deve ter uma versão da plataforma igual ou posterior à versão listada na tabela a seguir ou uma versão do Kubernetes posterior às versões listadas na tabela.

Versão do Kubernetes	Versão da plataforma
1.30	eks.2
1.29	eks.1
1.28	eks.6
1.27	eks.10
1.26	eks.11
1.25	eks.12
1.24	eks.15
1.23	eks.17

É possível verificar a versão atual do Kubernetes e da plataforma ao substituir *my-cluster* no comando a seguir pelo nome do seu cluster e executar o comando modificado: **aws eks describe-cluster --name *my-cluster* --query 'cluster.{"Kubernetes Version": version, "Platform Version": platformVersion}'**.

> **Important**
>
> Depois que o Amazon EKS atualiza o cluster para a versão da plataforma listada na tabela, ele cria uma entrada de acesso com permissões de administrador ao cluster para a entidade principal do IAM que originalmente criou o cluster. Se não quiser que a entidade principal do IAM tenha permissões de administrador para o cluster, remova a entrada de acesso criada pelo Amazon EKS.
>
> Para clusters com versões da plataforma anteriores às listadas na tabela anterior, o criador do cluster é sempre um administrador do cluster. Não é possível remover as permissões de administrador do cluster do usuário do IAM ou do perfil que criou o cluster.

- Uma entidade principal do IAM com as seguintes permissões para o cluster: `CreateAccessEntry`, `ListAccessEntries`, `DescribeAccessEntry`, `DeleteAccessEntry` e `UpdateAccessEntry`. Para obter mais informações sobre as permissões do Amazon EKS,

consulte Ações definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização do serviço.

- Uma entidade principal do IAM existente para criar uma entrada de acesso ou uma entrada de acesso existente para atualizar ou excluir.

Alterar o modo de autenticação para usar entradas de acesso

Para começar a usar entradas de acesso, você deve alterar o modo de autenticação do cluster para os modos `API_AND_CONFIG_MAP` ou `API`. Essa ação adiciona a API para entradas de acesso.

AWS Management Console

Para criar uma entrada de acesso

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha o nome do cluster em que você deseja criar uma entrada de acesso.
3. Escolha a guia Acesso.
4. O Modo de autenticação mostra o modo de autenticação atual do cluster. Se o modo indicar EKS API, já será possível adicionar entradas de acesso e pular as etapas restantes.
5. Selecione Gerenciar acesso.
6. Para o Modo de autenticação de cluster, selecione um modo com a EKS API. Observe que você não pode alterar o modo de autenticação de volta para um modo que remova as entradas da EKS API e de acesso.
7. Escolha Salvar alterações. O Amazon EKS começa a atualizar o cluster, o status do cluster muda para Updating e a alteração é registrada na guia Histórico de atualizações.
8. Aguarde até que o status do cluster retorne para Active. Quando o cluster estiver Active, será possível seguir as etapas em Criar entradas de acesso para adicionar acesso ao cluster para entidades principais do IAM.

AWS CLI

Pré-requisito

Instalar a AWS CLI conforme descrito em Instalar, atualizar e desinstalar a AWS CLI no Guia do usuário da AWS Command Line Interface.

1. Execute o seguinte comando . Substitua *my-cluster* pelo nome do cluster. Se você quiser desativar o método `ConfigMap` permanentemente, substitua `API_AND_CONFIG_MAP` por `API`.

   O Amazon EKS começa a atualizar o cluster, o status do cluster muda para UPDATING e a alteração é registrada em aws eks list-updates.

   ```
   aws eks update-cluster-config --name my-cluster --access-config
    authenticationMode=API_AND_CONFIG_MAP
   ```

2. Aguarde até que o status do cluster retorne para Active. Quando o cluster estiver Active, será possível seguir as etapas em Criar entradas de acesso para adicionar acesso ao cluster para entidades principais do IAM.

Criar entradas de acesso

Considerações

Antes de criar entradas de acesso, considere o seguinte:

- Um modo de autenticação definido corretamente. Consulte Alterar o modo de autenticação para usar entradas de acesso.
- Uma entrada de acesso inclui o nome do recurso da Amazon (ARN) de uma, e somente uma, entidade principal do IAM existente. Uma entidade principal do IAM não pode ser incluída em mais de uma entrada de acesso. Considerações adicionais para o ARN especificado por você:
  - As práticas recomendadas do IAM sugerem que você acesse o cluster usando perfis do IAM que têm credenciais de curto prazo, em vez de usuários do IAM que têm credenciais de longo prazo. Para obter mais informações, consulte Exija que os usuários humanos usem a federação com um provedor de identidades para acessar a AWS usando credenciais temporárias no Guia do usuário do IAM.
  - Se o ARN for para um perfil do IAM, ele pode incluir um caminho. Os ARNs nas entradas `ConfigMap` `aws-auth` não podem incluir um caminho. Por exemplo, seu ARN pode ser `arn:aws:iam::111122223333:role/development/apps/my-role` ou `arn:aws:iam::111122223333:role/my-role`.
  - Se o tipo da entrada de acesso for diferente de `STANDARD` (consulte a próxima consideração sobre os tipos), o ARN deverá estar na mesma Conta da AWS em que seu cluster está. Se o

tipo for `STANDARD`, o ARN poderá estar na mesma Conta da AWS, ou outra diferente, da conta em que seu cluster está.

- Não é possível alterar a entidade principal do IAM depois que a entrada de acesso é criada.
- Se você alguma vez você excluir a entidade principal do IAM com esse ARN, a entrada de acesso não será excluída automaticamente. Recomendamos que você exclua a entrada de acesso com um ARN para uma entidade principal do IAM que você exclua. Se você não excluir a entrada de acesso e nunca recriar a entidade principal do IAM, mesmo que ela tenha o mesmo ARN, a entrada de acesso não funcionará. Isto ocorre porque, embora o ARN seja o mesmo para a entidade principal do IAM recriado, o `roleID` ou `userID` (você pode ver isso com o comando `aws sts get-caller-identity` da AWS CLI) é diferente para a entidade principal do IAM recriada do que foi para a entidade principal do IAM original. Mesmo que você não veja o `roleID` ou `userID` da entidade principal do IAM para uma entrada de acesso, o Amazon EKS o armazena com a entrada de acesso.

- Cada entrada de acesso tem um tipo. Você pode especificar `EC2 Linux` (para um perfil do IAM usado com nós autogerenciados do Linux ou do Bottlerocket), `EC2 Windows` (para perfis do IAM usados com nós autogerenciados do Windows), `FARGATE_LINUX` (para perfis do IAM usados com AWS Fargate (Fargate)) ou `STANDARD` como um tipo. Se você não especificar um tipo, o Amazon EKS definirá automaticamente o tipo como `STANDARD`. Não é necessário criar uma entrada de acesso para um perfil do IAM usado para um grupo de nós gerenciados ou um perfil do Fargate, porque o Amazon EKS adiciona entradas para essas funções ao `ConfigMap` `aws-auth`, independentemente da versão da plataforma em que seu cluster esteja.

  Não é possível alterar o tipo depois que a entrada de acesso é criada.
- Se o tipo da entrada de acesso for `STANDARD`, será possível especificar um nome de usuário para a entrada de acesso. Se você não especificar um valor para o nome de usuário, o Amazon EKS definirá um dos valores a seguir para você, dependendo do tipo de entrada de acesso e de se a entidade principal do IAM que você especificou é um perfil do IAM ou um usuário do IAM. A menos que você tenha um motivo específico para determinar seu próprio nome de usuário, recomendamos que não especifique um e deixe o Amazon EKS gerá-lo automaticamente para você. Se você especificar seu próprio nome de usuário:
  - Não pode começar com `system:`, `eks:`, `aws:`, `amazon:` ou `iam:`.
  - Se o nome de usuário for para um perfil do IAM, recomendamos que você adicione `{{SessionName}}` ao final do seu nome de usuário. Se você adicionar `{{SessionName}}` ao seu nome de usuário, ele deverá incluir dois pontos antes de {{SessionName}}. Quando essa função é assumida, o nome da sessão especificado ao assumir a função é automaticamente

passado para o cluster e aparecerá nos logs do CloudTrail. Por exemplo, não é possível ter um nome de usuário `john{{SessionName}}`. O nome de usuário teria que ser `:john{{SessionName}}` ou `jo:hn{{SessionName}}`. Os dois pontos precisam vir antes de `{{SessionName}}`. O nome de usuário gerado pelo Amazon EKS na tabela a seguir inclui um ARN. Como um ARN inclui dois pontos, ele atende a esse requisito. Os dois pontos não serão necessários se você não incluir `{{SessionName}}` no seu nome de usuário.

Tipo de entidade principal do IAM	Tipo	Valor do nome de usuário que o Amazon EKS define automaticamente
Usuário	`STANDARD`	O ARN do usuário. Exemplo: `arn:aws:iam::` *`111122223333`* `:user/`*`my-user`*
Função	`STANDARD`	O STS ARN da função quando ela é assumida. O Amazon EKS anexa `{{SessionName}}` à função. Exemplo: `arn:aws:sts::` *`111122223333`* `:assumed-role/`*`my-role`*`/{{SessionName}}` Se o ARN da função que você especificou contiver um caminho, o Amazon EKS o removerá no nome de usuário gerado.
Função	`EC2 Linux` ou `EC2 Windows`	`system:node:{{EC2PrivateDNSName}}`

Tipo de entidade principal do IAM	Tipo	Valor do nome de usuário que o Amazon EKS define automaticamente
Função	`FARGATE_LINUX`	`system:node:{{SessionName}}`

Você poderá alterar o nome de usuário depois que a entrada de acesso for criada.

- Se o tipo de entrada de acesso for `STANDARD` e você quiser usar a autorização RBAC do Kubernetes, poderá adicionar um ou mais nomes de grupos à entrada de acesso. Depois de criar uma entrada de acesso, você pode adicionar e remover nomes de grupos. Para que a entidade principal do IAM tenha acesso aos objetos do Kubernetes em seu cluster, você deve criar e gerenciar objetos de autorização baseada em funções (RBAC) do Kubernetes. Crie objetos `RoleBinding` ou `ClusterRoleBinding` do Kubernetes em seu cluster que especifiquem o nome do grupo como um `subject` para `kind: Group`. O Kubernetes autoriza o acesso da entidade principal do IAM a qualquer objeto de cluster que você especificou em um objeto `Role` `ClusterRole` do Kubernetes que você também especificou no `roleRef` da vinculação. Se você especificar nomes de grupo, convém familiarizar-se com os objetos de autorização com base em função (RBAC) do Kubernetes. Para mais informações, consulte Using RBAC Authorization (Usar autorização RBAC) na documentação do Kubernetes.

  > **Important**
  >
  > O Amazon EKS não confirma que nenhum objeto RBAC do Kubernetes existente no seu cluster inclua nenhum dos nomes de grupos que você especificar.

  Em vez de, ou além de, o Kubernetes autorizar o acesso da entidade principal do IAM aos objetos do Kubernetes em seu cluster, você pode associar as políticas de acesso do Amazon EKS a uma entrada de acesso. O Amazon EKS autoriza as entidades principais do IAM a acessar objetos do Kubernetes em seu cluster com as permissões na política de acesso. Você pode definir o escopo das permissões de uma política de acesso aos namespaces do Kubernetes que você especificar. O uso de políticas de acesso não exige que você gerencie objetos RBAC do Kubernetes. Para obter mais informações, consulte Associar políticas de acesso a entradas de acesso.
- Se você criar uma entrada de acesso com o tipo `EC2 Linux` ou `EC2 Windows`, a entidade principal do IAM que criou a entrada de acesso deverá ter a permissão `iam:PassRole`. Para

obter mais informações, consulte Conceder permissões a um usuário para passar uma função para um Serviço da AWS no Guia do usuário do IAM.

- Semelhante ao comportamento padrão do IAM, a criação e as atualizações de entradas de acesso acabam sendo consistentes e podem levar alguns segundos para serem efetivadas após o retorno bem-sucedido da chamada inicial de API. Suas aplicações devem ser projetadas para levar em conta esses possíveis atrasos. Recomendamos que você não inclua criações ou atualizações de entrada de acesso nos caminhos de código críticos de alta disponibilidade da sua aplicação. Em vez disso, faça alterações do em uma rotina de inicialização ou de configuração separada que você executa com menos frequência. Além disso, certifique-se de verificar se as alterações foram propagadas antes que os fluxos de trabalho de produção dependam delas.
- As entradas de acesso não oferecem suporte para perfis vinculados ao serviço. Não é possível criar entradas de acesso nas quais o ARN da entidade principal corresponde a um perfil vinculado ao serviço. É possível identificar perfis vinculados ao serviço pelo ARN deles, que está no formato `arn:aws:iam::*:role/aws-service-role/*`.

Você pode criar uma entrada de acesso usando o AWS Management Console ou a AWS CLI.

AWS Management Console

Para criar uma entrada de acesso

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha o nome do cluster em que você deseja criar uma entrada de acesso.
3. Escolha a guia Acesso.
4. Selecione Criar entrada de acesso.
5. Para a entidade principal do IAM, selecione um perfil ou usuário do IAM existente. As práticas recomendadas do IAM sugerem que você acesse o cluster usando perfis do IAM que têm credenciais de curto prazo, em vez de usuários do IAM que têm credenciais de longo prazo. Para obter mais informações, consulte Exija que os usuários humanos usem a federação com um provedor de identidades para acessar a AWS usando credenciais temporárias no Guia do usuário do IAM.
6. Em Tipo, se a entrada de acesso for para a função de nó usada para nós autogerenciados do Amazon EC2, selecione EC2 Linux ou EC2 Windows. Caso contrário, aceite o padrão (Padrão).
7. Se o Tipo escolhido for Padrão e você quiser especificar um nome de usuário, insira o nome de usuário.

8. Se o Tipo escolhido for Padrão e você quiser usar a autorização RBAC do Kubernetes para a entidade principal do IAM, especifique um ou mais nomes para Grupos. Se você não especificar nenhum nome de grupo e quiser usar a autorização do Amazon EKS, poderá associar uma política de acesso em uma etapa posterior ou após a criação da entrada de acesso.
9. (Opcional) Para Tags, atribua rótulos à entrada de acesso. Por exemplo, para facilitar a localização de todos os recursos com a mesma tag.
10. Escolha Próximo.
11. Na página Adicionar política de acesso, se o tipo escolhido for Padrão e você quiser que o Amazon EKS autorize a entidade principal do IAM a ter permissões para os objetos do Kubernetes em seu cluster, execute as etapas a seguir. Caso contrário, escolha Next.

    a. Em Nome da política, escolha uma política de acesso. Você não pode ver as permissões das políticas de acesso, mas elas incluem permissões semelhantes às dos objetos `ClusterRole` voltados para o usuário do Kubernetes. Para obter mais informações, consulte User-facing roles na documentação do Kubernetes.

    b. Escolha uma das seguintes opções:

       - Cluster: escolha essa opção se quiser que o Amazon EKS autorize a entidade principal do IAM a ter as permissões na política de acesso para todos os objetos do Kubernetes em seu cluster.
       - Namespace do Kubernetes: escolha essa opção se quiser que o Amazon EKS autorize a entidade principal do IAM a ter as permissões na política de acesso para todos os objetos do Kubernetes em um namespace do Kubernetes específico em seu cluster. Em Namespace, insira o nome do namespace do Kubernetes no seu cluster. Se quiser adicionar mais namespaces, escolha Adicionar novo namespace e insira o nome do namespace.

    c. Se você quiser adicionar mais políticas, escolha Adicionar política. Você pode definir o escopo de cada política de forma diferente, mas cada política só pode ser adicionada uma vez.

    d. Escolha Próximo.
12. Revise a configuração da entrada de acesso. Se algo parecer errado, escolha Anterior para voltar às etapas anteriores e corrigir o erro. Se a configuração estiver correta, escolha Criar.

AWS CLI

Pré-requisito

Instalar a AWS CLI conforme descrito em Instalar, atualizar e desinstalar a AWS CLI no Guia do usuário da AWS Command Line Interface.

Para criar uma entrada de acesso

Você pode usar qualquer um dos seguintes exemplos para criar entradas de acesso:

- Crie uma entrada de acesso para um grupo de nós autogerenciado do Amazon EC2 Linux. Substitua *my-cluster* pelo nome do seu cluster, *111122223333* pelo seu ID da Conta da AWS e *EKS-my-cluster-self-managed-ng-1* pelo nome do seu nó do perfil do IAM. Se seu grupo de nós for um grupo de nós do Windows, substitua *EC2_Linux* por **EC2_Windows**.

  ```
  aws eks create-access-entry --cluster-name my-cluster --principal-arn
   arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1 --type EC2_Linux
  ```

  Você não pode usar a opção `--kubernetes-groups` ao especificar um tipo diferente de `STANDARD`. Você não pode associar uma política de acesso a essa entrada de acesso porque seu tipo é um valor diferente de `STANDARD`.
- Crie uma entrada de acesso que permita um perfil do IAM que não é usado para um grupo de nós autogerenciado do Amazon EC2, com o qual você deseja que o Kubernetes autorize o acesso ao seu cluster. Substitua *my-cluster* pelo nome do cluster e *111122223333* pelo seu ID da Conta da AWS e *my-role* pelo nome do seu perfil do IAM. Substitua *Espectadores* pelo nome de um grupo que você especificou em um objeto `RoleBinding` ou `ClusterRoleBinding` do Kubernetes em seu cluster.

  ```
  aws eks create-access-entry --cluster-name my-cluster --principal-arn
   arn:aws:iam::111122223333:role/my-role --type STANDARD --user Viewers --
  kubernetes-groups Viewers
  ```
- Crie uma entrada de acesso que permita que um usuário do IAM se autentique no seu cluster. Esse exemplo é fornecido porque isso é possível, embora as práticas recomendadas do IAM sugiram acessar seu cluster usando perfis do IAM que têm credenciais de curto prazo, em vez de usuários do IAM que têm credenciais de longo prazo. Para obter mais informações, consulte Exija que os usuários humanos usem a federação com um provedor de identidades para acessar a AWS usando credenciais temporárias no Guia do usuário do IAM.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:user/my-user --type STANDARD --username my-user
```

Se quiser que esse usuário tenha mais acesso ao seu cluster do que as permissões nas funções de descoberta da API do Kubernetes, você precisará associar uma política de acesso à entrada de acesso, já que a opção `--kubernetes-groups` não é usada. Para obter mais informações, consulte Associar políticas de acesso a entradas de acesso e API discovery roles na documentação do Kubernetes.

Atualizar entradas de acesso

Você pode atualizar uma entrada de acesso usando o AWS Management Console ou a AWS CLI.

AWS Management Console

Para atualizar uma entrada de acesso

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha o nome do cluster em que você deseja criar uma entrada de acesso.
3. Escolha a guia Acesso.
4. Selecione a entrada de acesso que você deseja atualizar.
5. Selecione a opção Editar.
6. Em Nome de usuário, você pode alterar o valor existente.
7. Em Grupos, você pode remover nomes de grupos existentes ou adicionar novos nomes de grupos. Se os seguintes nomes de grupos existirem, não os remova: system:nodes ou system:bootstrappers. A remoção desses grupos pode fazer com que seu cluster funcione incorretamente. Se você não especificar nenhum nome de grupo e quiser usar a autorização do Amazon EKS, associe uma política de acesso em uma etapa posterior.
8. Em Tags, você pode atribuir rótulos à entrada de acesso. Por exemplo, para facilitar a localização de todos os recursos com a mesma tag. Você também pode remover as tags existentes.
9. Escolha Salvar alterações.
10. Se você quiser associar uma política de acesso à entrada, consulte Associar políticas de acesso a entradas de acesso.

AWS CLI

Pré-requisito

Instalar a AWS CLI conforme descrito em Instalar, atualizar e desinstalar a AWS CLI no Guia do usuário da AWS Command Line Interface.

Para atualizar uma entrada de acesso

Substitua *my-cluster* pelo nome do seu cluster, *111122223333* pelo seu ID da Conta da AWS e *EKS-my-cluster-my-namespace-Viewers* pelo nome de um perfil do IAM.

```
aws eks update-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --kubernetes-
groups Viewers
```

Você não poderá usar a opção `--kubernetes-groups` se o tipo da entrada de acesso for um valor diferente de `STANDARD`. Você também não pode associar uma política de acesso a uma entrada de acesso com um tipo diferente de `STANDARD`.

Excluir entradas de acesso

Se você descobrir que excluiu uma entrada de acesso por engano, sempre poderá recriá-la. Se a entrada de acesso que você está excluindo estiver associada a alguma política de acesso, as associações serão excluídas automaticamente. Você não precisa desassociar as políticas de acesso de uma entrada de acesso antes de excluir a entrada de acesso.

É possível excluir uma entrada de acesso usando o AWS Management Console ou a AWS CLI.

AWS Management Console

Para excluir uma chave de acesso

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha o nome do cluster do qual você deseja excluir uma entrada de acesso.
3. Escolha a guia Acesso.
4. Na lista Entradas de acesso, escolha a entrada de acesso que você deseja excluir.
5. Escolha Excluir.

6. Na caixa de diálogo de confirmação, escolha Excluir.

AWS CLI

Pré-requisito

Instalar a AWS CLI conforme descrito em Instalar, atualizar e desinstalar a AWS CLI no Guia do usuário da AWS Command Line Interface.

Para excluir uma chave de acesso

Substitua *my-cluster* pelo nome do seu cluster, *111122223333* pelo seu ID da Conta da AWS e *my-role* pelo nome do perfil do IAM que você não quer mais que tenha acesso ao seu cluster.

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

Associar políticas de acesso a entradas de acesso

Você pode atribuir uma ou mais políticas de acesso para acessar entradas do tipo `STANDARD`. O Amazon EKS concede automaticamente aos outros tipos de entradas de acesso as permissões necessárias para funcionar adequadamente em seu cluster. As políticas de acesso do Amazon EKS incluem permissões do Kubernetes, não permissões do IAM. Antes de associar uma política de acesso a uma entrada de acesso, familiarize-se com as permissões do Kubernetes incluídas em cada política de acesso. Para obter mais informações, consulte Permissão da política de acesso a dados. Se nenhuma das políticas de acesso atender aos seus requisitos, não associe uma política de acesso a uma entrada de acesso. Em vez disso, especifique um ou mais nomes de grupos para a entrada de acesso e crie e gerencie objetos de controle de acesso baseados em funções do Kubernetes. Para obter mais informações, consulte Criar entradas de acesso.

Pré-requisitos

- Uma entrada de acesso existente. Para criar uma, consulte Criar entradas de acesso.
- Uma função ou usuário do AWS Identity and Access Management com as seguintes permissões: `ListAccessEntries`, `DescribeAccessEntry`, `UpdateAccessEntry`, `ListAccessPolicies`, `AssociateAccessPolicy` e `DisassociateAccessPolicy`. Para obter mais informações, consulte Ações definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização do serviço.

Antes de associar políticas de acesso às entradas de acesso, considere os seguintes requisitos:

- Você pode associar várias políticas de acesso a cada entrada de acesso, mas só pode associar cada política a uma entrada de acesso uma vez. Se você associar várias políticas de acesso, a entidade principal do IAM da entrada de acesso tem todas as permissões incluídas em todas as políticas de acesso associadas.
- Você pode definir o escopo de uma política de acesso para todos os recursos em um cluster ou especificando o nome de um ou mais namespaces do Kubernetes. Você pode usar caracteres curinga para o nome de um namespace. Por exemplo, se você quiser definir o escopo de uma política de acesso para todos os namespaces que começam com `dev-`, você pode especificar `dev-*` como um nome de namespace. Verifique se os namespaces existem no cluster e se a ortografia corresponde ao nome real do namespace no cluster. O Amazon EKS não confirma a ortografia nem a existência dos namespaces no seu cluster.
- Você pode alterar o escopo de acesso de uma política de acesso depois de associá-la a uma entrada de acesso. Se você definiu o escopo da política de acesso para namespaces do Kubernetes, pode adicionar e remover namespaces da associação, conforme necessário.
- Se você associar uma política de acesso a uma entrada de acesso que também tenha nomes de grupos especificados, a entidade principal do IAM terá todas as permissões em todas as políticas de acesso associadas. Ela também tem todas as permissões em qualquer objeto `Role` ou `ClusterRole` do Kubernetes especificado em qualquer objeto `Role` ou `RoleBinding` do Kubernetes que especificam os nomes dos grupos.
- Se você executar o comando `kubectl auth can-i --list`, não verá nenhuma permissão Kubernetes atribuída pelas políticas de acesso associadas a uma entrada de acesso para a entidade principal do IAM que você está usando ao executar o comando. O comando só mostra permissões Kubernetes se você as concedeu em objetos `Role` ou `ClusterRole` do Kubernetes vinculados aos nomes de grupo ou nome de usuário que você especificou para uma entrada de acesso.
- Se você se faz passar por um usuário ou grupo do Kubernetes ao interagir com objetos do Kubernetes em seu cluster, como usar o comando `kubectl` com **`--as`** ***`username`*** ou **`--as-group`** ***`group-name`***, você está forçando o uso da autorização RBAC do Kubernetes. Como resultado, a entidade principal do IAM não tem permissões atribuídas por nenhuma política de acesso associada à entrada de acesso. As únicas permissões do Kubernetes que o usuário ou grupo que a entidade principal do IAM está representando tem são as permissões do Kubernetes que você concedeu a eles em objetos `Role` ou `ClusterRole` do Kubernetes que você vinculou aos nomes do grupo ou do usuário. Para que sua entidade principal do IAM tenha as permissões nas políticas de acesso associadas, não se faça passar por um usuário ou grupo do Kubernetes.

A entidade principal do IAM ainda terá todas as permissões que você concedeu a ele nos objetos `Role` ou `ClusterRole` do Kubernetes que você vinculou aos nomes de grupo ou nome de usuário que você especificou para a entrada de acesso. Para obter mais informações, consulte Personificação de usuário na documentação do Kubernetes.

Você pode associar uma política de acesso a uma entrada de acesso usando o AWS Management Console ou a AWS CLI.

AWS Management Console

Para associar uma política de acesso a uma entrada de acesso usando o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha o nome do cluster que tem uma entrada de acesso à qual você deseja associar uma política de acesso.
3. Escolha a guia Acesso.
4. Se o tipo da entrada de acesso for Padrão, você poderá associar ou desassociar as políticas de acesso do Amazon EKS. Se o tipo da sua entrada de acesso for diferente de Padrão, essa opção não estará disponível.
5. Escolha Associar política de acesso.
6. Em Nome da política, selecione a política com as permissões que você deseja que a entidade principal do IAM tenha.. Para visualizar as permissões incluídas em cada política, consulte Permissão da política de acesso a dados.
7. Em Escopo de acesso, escolha um escopo de acesso. Se você escolher Cluster, as permissões na política de acesso serão concedidas à entidade principal do IAM para recursos em todos os namespaces do Kubernetes. Se você escolher Namespace do Kubernetes, poderá então escolher Adicionar novo namespace. No campo Namespace exibido, você pode inserir o nome de um namespace do Kubernetes no seu cluster. Se quiser que a entidade principal do IAM tenha as permissões em vários namespaces, você pode inserir vários namespaces.
8. Escolha Adicionar política de acesso.

AWS CLI

Pré-requisito

A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.

Para associar uma política de acesso a uma entrada de acesso

1. Visualize as políticas de acesso disponíveis.

```
aws eks list-access-policies --output table
```

Veja um exemplo de saída abaixo.

```
-------------------------------------------------------------------------------------
|                                        ListAccessPolicies
                        |
+-----------------------------------------------------------------------------------
+
||                                          accessPolicies
                       ||
|+-------------------------------------------------------------------
+-------------------------------+|
||                                arn                                |
     name              ||
|+-------------------------------------------------------------------
+-------------------------------+|
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy       |
 AmazonEKSAdminPolicy         ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy |
 AmazonEKSClusterAdminPolicy  ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy        |
 AmazonEKSEditPolicy          ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy        |
 AmazonEKSViewPolicy          ||
```

```
|+-------------------------------------------------------------------
+------------------------------+|
```

Para visualizar as permissões incluídas em cada política, consulte Permissão da política de acesso a dados.

2. Visualize suas entradas de acesso existentes. Substitua *my-cluster* pelo nome do cluster.

```
aws eks list-access-entries --cluster-name my-cluster
```

Veja um exemplo de saída abaixo.

```
{
    "accessEntries": [
        "arn:aws:iam::111122223333:role/my-role",
        "arn:aws:iam::111122223333:user/my-user"
    ]
}
```

3. Associe uma política de acesso a uma entrada de acesso. O exemplo a seguir associa a política de acesso do AmazonEKSViewPolicy a uma entrada de acesso. Sempre que o perfil do IAM *my-role* tentar acessar objetos do Kubernetes no cluster, o Amazon EKS autorizará a função a usar as permissões na política para acessar objetos do Kubernetes somente nos namespaces *my-namespace1* e *my-namespace2* do Kubernetes. Substitua *my-cluster* pelo nome do seu cluster, *111122223333* pelo seu ID da Conta da AWS e *my-role* pelo nome do perfil do IAM para o qual você deseja que o Amazon EKS autorize o acesso aos objetos do cluster do Kubernetes.

```
aws eks associate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
   --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --
policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy
```

Se você quiser que a entidade principal do IAM tenha as permissões em todo o cluster, substitua **type=namespace,namespaces=*my-namespace1*,*my-namespace2*** por **type=cluster**. Se você quiser associar várias políticas de acesso à entrada de acesso, execute o comando várias vezes, cada uma com uma política de acesso exclusiva. Cada política de acesso associada tem seu próprio escopo.

> **Note**
>
> Se você quiser alterar posteriormente o escopo de uma política de acesso associada, execute o comando anterior novamente com o novo escopo. Por exemplo, se você quisesse remover *my-namespace2*, executaria o comando novamente usando somente **type=namespace,namespaces=*my-namespace1***. Se você quisesse alterar o escopo de **namespace** para **cluster**, executaria o comando novamente usando **type=cluster**, removendo **type=namespace,namespaces=*my-namespace1*,*my-namespace2***.

Para desassociar uma política de acesso de uma entrada de acesso

1. Determine quais políticas de acesso estão associadas a uma entrada de acesso.

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-
arn arn:aws:iam::111122223333:role/my-role
```

Veja um exemplo de saída abaixo.

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws:iam::111122223333",
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws:eks::aws:cluster-access-
policy/AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster",
                "namespaces": []
            },
            "associatedAt": "2023-04-17T15:25:21.675000-04:00",
            "modifiedAt": "2023-04-17T15:25:21.675000-04:00"
        },
        {
            "policyArn": "arn:aws:eks::aws:cluster-access-
policy/AmazonEKSAdminPolicy",
            "accessScope": {
                "type": "namespace",
                "namespaces": [
```

```
                    "my-namespace1",
                    "my-namespace2"
                ]
            },
            "associatedAt": "2023-04-17T15:02:06.511000-04:00",
            "modifiedAt": "2023-04-17T15:02:06.511000-04:00"
        }
    ]
}
```

No exemplo anterior, a entidade principal do IAM para essa entrada de acesso tem permissões de visualização em todos os namespaces no cluster e permissões de administrador em dois namespaces do Kubernetes.

2. Desassocie uma política de acesso de uma entrada de acesso. Neste exemplo, a política `AmazonEKSAdminPolicy` é desassociada de uma entrada de acesso. No entanto, a entidade principal do IAM retém as permissões na política de acesso `AmazonEKSViewPolicy` para objetos nos namespaces *my-namespace1* e *my-namespace2* porque essa política de acesso não está dissociada da entrada de acesso.

```
aws eks disassociate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
    --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy
```

Permissão da política de acesso a dados

As políticas de acesso incluem `rules` que contêm `verbs` (permissões) e `resources` do Kubernetes. As políticas de acesso não incluem permissões ou recursos do IAM. Semelhantes aos objetos `Role` e `ClusterRole` do Kubernetes, as políticas de acesso incluem apenas `rules` `allow`. Não é possível modificar o conteúdo de uma política de acesso. Você não pode criar suas próprias políticas de acesso. Se as permissões nas políticas de acesso não atenderem às suas necessidades, crie objetos RBAC do Kubernetes e especifique nomes de grupos para suas entradas de acesso. Para obter mais informações, consulte Criar entradas de acesso. As permissões contidas nas políticas de acesso são semelhantes às permissões nas funções de cluster do Kubernetes voltadas para o usuário. Para obter mais informações, consulte User-facing roles na documentação do Kubernetes.

Escolha qualquer política de acesso para ver seu conteúdo. Cada linha de cada tabela em cada política de acesso é uma regra separada.

AmazonEKSAdminPolicy

Essa política de acesso inclui permissões que concedem à entidade principal do IAM a maioria das permissões aos recursos. Quando associada a uma entrada de acesso, seu escopo de acesso geralmente é um ou mais namespaces do Kubernetes. Se você quiser que uma entidade principal do IAM tenha acesso de administrador a todos os recursos do seu cluster, associe a política de acesso AmazonEKSClusterAdminPolicy à sua entrada de acesso.

ARN: `arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy`

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
authorization.k8s.io	localsubjectaccessreviews	create
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
networking.k8s.io	ingresses , ingresses /status , networkpo licies	get, list, watch
networking.k8s.io	ingresses , networkpo licies	create, delete, deletecol lection , patch, update
policy	poddisruptionbudgets	create, delete, deletecol lection , patch, update
policy	poddisruptionbudgets , poddisruptionbudge ts/status	get, list, watch
rbac.authorization .k8s.io	rolebindings , roles	create, delete, deletecol lection , get, list, patch, update, watch
	configmaps , endpoints , persistentvolumecl aims , persisten tvolumeclaims/stat us , pods, replicati oncontrollers , replicationcontrol lers/scale , serviceac counts , services, services/status	get,list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update
	serviceaccounts	impersonate
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch
	namespaces	get,list, watch

AmazonEKSClusterAdminPolicy

Essa política de acesso inclui permissões que concedem ao administrador principal do IAM acesso a um cluster. Quando associada a uma entrada de acesso, seu escopo de acesso geralmente é o cluster, em vez de um namespace do Kubernetes. Se você quiser que uma entidade principal do IAM tenha um escopo administrativo mais limitado, considere associar a política de acesso AmazonEKSAdminPolicy à sua entrada de acesso.

ARN: `arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy`

Grupos da API do Kubernetes	nonResourceURLs do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
*		*	*
	*		*

AmazonEKSAdminViewPolicy

Essa política de acesso inclui permissões que concedem a uma entidade principal do IAM acesso de listagem/visualização a todos os recursos em um cluster. Observe que isso inclui Segredos do Kubernetes.

ARN: `arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy`

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
*	*	`get, list, watch`

AmazonEKSEditPolicy

Essa política de acesso inclui permissões para que uma entidade principal do IAM edite a maioria dos recursos do Kubernetes.

ARN: `arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy`

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
`apps`	`daemonsets` , `deployments` , `deployments/rollback` , `deployments/scale` , `replicasets` , `replicasets/`	`create`, `delete`, `deletecollection` , `patch`, `update`

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
	scale, statefulsets , statefulsets/scale	
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
	namespaces	get, list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	serviceaccounts	impersonate
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
	`bindings`, `events`, `limitranges` , `namespaces/status` , `pods/log`, `pods/status` , `replicationcontrollers/status` , `resourcequotas` , `resourcequotas/status`	`get`, `list`, `watch`

AmazonEKSViewPolicy

Essa política de acesso inclui permissões para que uma entidade principal do IAM visualize a maioria dos recursos do Kubernetes.

ARN: `arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy`

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
`apps`	`controllerrevisions` , `daemonsets` , `daemonsets/status` , `deployments` , `deployments/scale` , `deployments/status` , `replicasets` , `replicasets/scale` , `replicasets/status` , `statefulsets` , `statefulsets/scale` , `statefulsets/status`	`get`, `list`, `watch`
`autoscaling`	`horizontalpodautoscalers` , `horizonta`	`get`, `list`, `watch`

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
	lpodautoscalers/status	
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch

Grupos da API do Kubernetes	Recursos Kubernetes	Verbos (permissões) do Kubernetes
	`configmaps` , `endpoints` , `persistentvolumeclaims` , `persistentvolumeclaims/status` , `pods`, `replicationcontrollers` , `replicationcontrollers/scale` , `serviceaccounts` , `services`, `services/status`	`get`, `list`, `watch`
	`bindings`, `events`, `limitranges` , `namespaces/status` , `pods/log`, `pods/status` , `replicationcontrollers/status` , `resourcequotas` , `resourcequotas/status`	`get`, `list`, `watch`
	`namespaces`	`get`, `list`, `watch`

Atualizações de política de acesso

Visualize detalhes sobre as atualizações das políticas de acesso, desde que elas foram introduzidas. Para receber alertas automáticos sobre alterações nesta página, assine o feed RSS na página Histórico do documento do Amazon EKS.

Alteração	Descrição	Data
Adicionar `AmazonEKSAdminViewPolicy`	Adicione uma nova política para acesso de visualização expandido, incluindo recursos como Segredos.	23 de abril de 2024

Alteração	Descrição	Data
Políticas de acesso introduzidas.	O Amazon EKS introduziu políticas de acesso.	29 de maio de 2023

Migrando entradas existentes `aws-auth ConfigMap` para entradas de acesso

Se você adicionou entradas ao `ConfigMap` `aws-auth` no seu cluster, recomendamos que você crie entradas de acesso para as entradas existentes no seu `ConfigMap` `aws-auth`. Depois de criar as entradas de acesso, você pode remover as entradas do seu `ConfigMap`. Você não pode associar políticas de acesso às entradas no `ConfigMap` `aws-auth`. Se você quiser associar políticas de acesso às entidades principais do IAM, crie entradas de acesso.

> **Important**
>
> Não remova as entradas `ConfigMap` `aws-auth` existentes que foram criadas pelo Amazon EKS quando você adicionou um grupo de nós gerenciados ou um perfil do Fargate ao seu cluster. Se você remover as entradas que o Amazon EKS criou no `ConfigMap`, seu cluster não funcionará corretamente. No entanto, você pode remover todas as entradas dos grupos de nós autogerenciados depois de criar entradas de acesso para eles.

Pré-requisitos

- Familiaridade com entradas de acesso e políticas de acesso. Para ter mais informações, consulte Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS e Associar políticas de acesso a entradas de acesso.
- Um cluster existente com uma versão da plataforma igual ou posterior às versões listadas nos Pré-requisitos do tópico Permitir que perfis ou usuários do IAM acessem objetos do Kubernetes em seu cluster do Amazon EKS.
- Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
- Permissões do Kubernetes para modificar o `aws-auth` `ConfigMap` no namespace `kube-system`.

- Uma função do AWS Identity and Access Management ou usuário com as seguintes permissões: `CreateAccessEntry` e `ListAccessEntries`. Para obter mais informações, consulte Ações definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização do serviço.

Para migrar uma entrada de seu **`aws-auth ConfigMap`** para uma entrada de acesso

1. Veja as entradas existentes em seu `aws-auth ConfigMap`. Substitua *`my-cluster`* pelo nome do cluster.

   ```
   eksctl get iamidentitymapping --cluster my-cluster
   ```

   Veja um exemplo de saída abaixo.

   ```
   ARN
                USERNAME                                GROUPS
                              ACCOUNT
   arn:aws:iam::111122223333:role/EKS-my-cluster-Admins
                Admins                                  system:masters
   arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers
                my-namespace-Viewers                    Viewers
   arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1
                               system:node:{{EC2PrivateDNSName}}
    system:bootstrappers,system:nodes
   arn:aws:iam::111122223333:user/my-user
                my-user
   arn:aws:iam::111122223333:role/EKS-my-cluster-fargateprofile1
                               system:node:{{SessionName}}
    system:bootstrappers,system:nodes,system:node-proxier
   arn:aws:iam::111122223333:role/EKS-my-cluster-managed-ng
                               system:node:{{EC2PrivateDNSName}}
    system:bootstrappers,system:nodes
   ```

2. Crie entradas de acesso para qualquer uma das entradas do `ConfigMap` que você criou retornadas na saída anterior. Ao criar as entradas de acesso, verifique se você especificou os mesmos valores para`ARN`, `USERNAME`, `GROUPS` e `ACCOUNT` retornados em sua saída. No exemplo de saída, você criaria entradas de acesso para todas as entradas, exceto as duas últimas, já que essas entradas foram criadas pelo Amazon EKS para um perfil do Fargate e um grupo de nós gerenciados.
3. Exclua as entradas do `ConfigMap` para todas as entradas de acesso que você criou. Se você não excluir a entrada do `ConfigMap`, as configurações da entrada de acesso do ARN principal

do IAM substituirão a entrada do `ConfigMap`. Substitua *111122223333* pelo seu ID da Conta da AWS e *EKS-my-cluster-my-namespace-Viewers* pelo nome da função na entrada do seu `ConfigMap`. Se a entrada que você está removendo for para um usuário do IAM, em vez de um perfil do IAM, substitua **role** por **user** e *EKS-my-cluster-my-namespace-Viewers* pelo nome do usuário.

```
eksctl delete iamidentitymapping --arn arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --cluster my-cluster
```

Conceder aos usuários do IAM acesso ao Kubernetes com um ConfigMap

> **Important**
>
> O ConfigMap `aws-auth` está obsoleto. O método recomendado para gerenciar o acesso às APIs do Kubernetes é Entradas de acesso.

O acesso ao cluster usando as entidades do IAM é habilitado pelo AWS IAM Authenticator para Kubernetes, que é executado no ambiente de gerenciamento do Amazon EKS. O autenticador obtém suas informações de configuração do `ConfigMap` `aws-auth`. Para todas as configurações do `ConfigMap` do `aws-auth`, consulte Full Configuration Format (Formato de configuração completo) no GitHub.

Adicione um usuário do IAM ao cluster do Amazon EKS

Quando você cria um cluster do Amazon EKS, a entidade principal do IAM que cria o cluster, recebe automaticamente permissões `system:masters` na configuração de controle de acesso baseado em perfil (RBAC) no ambiente de gerenciamento do Amazon EKS. Como essa entidade principal não é exibida em nenhuma configuração visível, mantenha o controle de qual entidade principal criou o cluster originalmente. Para conceder a outras entidades principais do IAM a capacidade de interagir com o cluster, edite o `aws-auth` `ConfigMap` no Kubernetes e crie um Kubernetes `rolebinding` ou `clusterrolebinding` com o nome de um `group` que você especifica no `aws-auth` `ConfigMap`.

> **Note**
>
> Para obter mais informações sobre a configuração de controle de acesso baseado em função (RBAC) do Kubernetes, consulte Using RBAC Authorization (Usar autorização RBAC) na documentação do Kubernetes.

Para adicionar uma entidade principal do IAM ao cluster do Amazon EKS

1. Determine quais credenciais o `kubectl` está usando para acessar o cluster. Em seu computador, você pode ver quais credenciais o `kubectl` usa com o comando a seguir. Substitua ***~/.kube/config*** pelo caminho para o arquivo `kubeconfig`, se você não usar o caminho padrão.

   ```
   cat ~/.kube/config
   ```

   Veja um exemplo de saída abaixo.

   ```
   [...]
   contexts:
   - context:
       cluster: my-cluster.region-code.eksctl.io
       user: admin@my-cluster.region-code.eksctl.io
     name: admin@my-cluster.region-code.eksctl.io
   current-context: admin@my-cluster.region-code.eksctl.io
   [...]
   ```

   No exemplo de saída anterior, as credenciais de um usuário denominado *`admin`* são configuradas para um cluster denominado *`my-cluster`*. Se for o usuário que criou o cluster, ele já terá acesso ao cluster. Se não for o usuário que criou o cluster, será necessário concluir as etapas restantes para habilitar o acesso ao cluster para os outras entidades principais do IAM. As melhores práticas do IAM recomendam que você conceda permissões para perfis e não para usuários. É possível ver quais outras entidades principais atualmente têm acesso ao cluster com o seguinte comando:

   ```
   kubectl describe -n kube-system configmap/aws-auth
   ```

   Veja um exemplo de saída abaixo.

```
Name:         aws-auth
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws:iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}


BinaryData
====

Events:  <none>
```

O exemplo anterior é um `aws-auth` `ConfigMap` padrão. Apenas a função de instância de nó tem acesso ao cluster.

2. Verifique se você tem as `roles` e `rolebindings` ou `clusterroles` e `clusterrolebindings` existentes do Kubernetes para as quais é possível mapear as entidades principais do IAM. Para saber mais sobre esses recursos, consulte Using RBAC Authorization (Usar autorização RBAC) na documentação do Kubernetes.

   1. Visualize as `roles` e `clusterroles` existentes do Kubernetes. As `Roles` têm o escopo definido para um `namespace`, mas `clusterroles` têm o escopo definido para o cluster.

      ```
      kubectl get roles -A
      ```

      ```
      kubectl get clusterroles
      ```

   2. Visualize os detalhes de qualquer `role` ou `clusterrole` retornada no resultado anterior e confirme que ela tem as permissões (`rules`) que você quer que as entidades principais do IAM tenham no cluster.

Substitua *role-name* por um nome de `role` retornado na saída do comando anterior. Substitua *kube-system* pelo namespace da `role`.

```
kubectl describe role role-name -n kube-system
```

Substitua *cluster-role-name* por um nome de `clusterrole` retornado na saída do comando anterior.

```
kubectl describe clusterrole cluster-role-name
```

3. Visualize as `rolebindings` e `clusterrolebindings` existentes do Kubernetes. As `Rolebindings` têm o escopo definido para um `namespace`, mas `clusterrolebindings` têm o escopo definido para o cluster.

```
kubectl get rolebindings -A
```

```
kubectl get clusterrolebindings
```

4. Visualize os detalhes de qualquer `rolebinding` ou `clusterrolebinding` e confirme que ele tem uma `role` ou `clusterrole` da etapa anterior listada como `roleRef` e um nome de grupo listado para `subjects`.

   Substitua *role-binding-name* por um nome de `rolebinding` retornado na saída do comando anterior. Substitua *kube-system* pelo `namespace` de `rolebinding`.

```
kubectl describe rolebinding role-binding-name -n kube-system
```

   Veja um exemplo de saída abaixo.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eks-console-dashboard-restricted-access-role-binding
  namespace: default
subjects:
- kind: Group
  name: eks-console-dashboard-restricted-access-group
  apiGroup: rbac.authorization.k8s.io
```

```
roleRef:
  kind: Role
  name: eks-console-dashboard-restricted-access-role
  apiGroup: rbac.authorization.k8s.io
```

Substitua *cluster-role-binding-name* por um nome de `clusterrolebinding` retornado na saída do comando anterior.

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

Veja um exemplo de saída abaixo.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
- kind: Group
  name: eks-console-dashboard-full-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: eks-console-dashboard-full-access-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

3. Edite o `ConfigMap` `aws-auth`. É possível usar uma ferramenta como o `eksctl` para atualizar o `ConfigMap`, ou você pode atualizá-lo manualmente via edição.

> **Important**
>
> Convém usar `eksctl` ou outra ferramenta para editar o `ConfigMap`. Para saber mais sobre outras ferramentas que podem ser usadas, consulte Usar ferramentas para fazer alterações em `aws-auth` `ConfigMap` nos guias de práticas recomendadas do Amazon EKS. Um `aws-auth` `ConfigMap` formatado incorretamente pode fazer com que você perca o acesso ao cluster.

eksctl

Pré-requisito

Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

1. Visualize os mapeamentos atuais no `ConfigMap`. Substitua o *`my-cluster`* pelo nome do cluster. Substitua *`region-code`* pela Região da AWS em que está o cluster.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

   Veja um exemplo de saída abaixo.

```
ARN
                    USERNAME                                GROUPS
             ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

2. Adicione um mapeamento referente a uma função. Substitua *`my-role`* pelo nome da função. Substitua *`eks-console-dashboard-full-access-group`* pelo nome do grupo especificado no objeto `RoleBinding` ou `ClusterRoleBinding` do Kubernetes. Substitua *`111122223333`* pelo ID da sua conta. Você pode substituir *`admin`* por qualquer nome que escolher.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws:iam::111122223333:role/my-role --username admin --group eks-
console-dashboard-full-access-group \
    --no-duplicate-arns
```

   > **Important**
   >
   > O ARN da função não pode incluir um caminho, como `role/my-team/developers/my-role`. O formato do ARN deve ser `arn:aws:iam::111122223333:role/my-role`. Neste exemplo, `my-team/developers/` precisa ser removido.

   Veja um exemplo de saída abaixo.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "arn:aws:iam::111122223333:role/my-
role" to auth ConfigMap
```

3. Adicione um mapeamento referente a um usuário. As melhores práticas do IAM recomendam que você conceda permissões para perfis e não para usuários. Substitua *my-user* pelo nome do usuário. Substitua *eks-console-dashboard-restricted-access-group* pelo nome do grupo especificado no objeto `RoleBinding` ou `ClusterRoleBinding` do Kubernetes. Substitua *111122223333* pelo ID da sua conta. Você pode substituir *my-user* por qualquer nome que escolher.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user --username my-user --
group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

Veja um exemplo de saída abaixo.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws:iam::111122223333:user/my-
user" to auth ConfigMap
```

4. Visualize os mapeamentos no `ConfigMap` novamente.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

Veja um exemplo de saída abaixo.

```
ARN
                  USERNAME                                GROUPS
                    ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:role/admin
                      my-role                                 eks-console-
dashboard-full-access-group
```

```
arn:aws:iam::111122223333:user/my-user
                        my-user                                    eks-console-
dashboard-restricted-access-group
```

Edit ConfigMap manually

1. Abra o `ConfigMap` para edição.

```
kubectl edit -n kube-system configmap/aws-auth
```

> **Note**
>
> Se você receber um erro informando "`Error from server (NotFound): configmaps "aws-auth" not found`", siga o procedimento em Como aplicar o `ConfigMap`aws-auth ao seu cluster para aplicar o `ConfigMap` comum.

2. Adicione as entidades principais do IAM ao `ConfigMap`. Um grupo do IAM não é uma entidade principal do IAM, portanto, não pode ser adicionado ao `ConfigMap`.
   - Para adicionar uma função do IAM (por exemplo, para usuários federados): adicione os detalhes da função à seção `mapRoles` do `ConfigMap` em `data`. Adicione essa seção se ela ainda não existir no arquivo. Cada entrada oferece suporte aos seguintes parâmetros:
     - rolearn: o ARN da função do IAM a ser adicionada. Esse valor não pode incluir um caminho. Por exemplo, não é possível especificar um ARN como `arn:aws:iam::111122223333:role/my-team/developers/role-name`. O ARN precisa ser `arn:aws:iam::111122223333:role/role-name`.
     - username: o nome do usuário no Kubernetes a ser mapeado para o perfil do IAM.
     - groups: o grupo ou a lista de grupos do Kubernetes para o qual mapear a função. O grupo pode ser padrão ou um grupo especificado em um `clusterrolebinding` ou `rolebinding`. Para obter mais informações, consulte Perfis padrão e vinculações de perfis na documentação do Kubernetes.
   - Para adicionar um usuário do IAM: as melhores práticas do IAM recomendam que você conceda permissões para perfis e não para usuários. Adicione os detalhes do usuário à seção `mapUsers` do `ConfigMap` em `data`. Adicione essa seção se ela ainda não existir no arquivo. Cada entrada oferece suporte aos seguintes parâmetros:

- userarn: o ARN do usuário do IAM a ser adicionado.
- username (nomeusuário): o nome do usuário no Kubernetes a ser mapeado para o usuário do IAM.
- groups (grupos): o grupo ou a lista de grupos do Kubernetes para o qual mapear o usuário. O grupo pode ser padrão ou um grupo especificado em um `clusterrolebinding` ou `rolebinding`. Para obter mais informações, consulte Perfis padrão e vinculações de perfis na documentação do Kubernetes.

Por exemplo, o seguinte bloco YAML contém:

- Uma seção `mapRoles` que mapeia a instância de nó do IAM para grupos do Kubernetes para que os nós possam se registrar no cluster e o perfil do IAM `my-console-viewer-role` que é mapeado para um grupo do Kubernetes que pode visualizar todos os recursos do Kubernetes para todos os clusters. Para obter uma lista das permissões de grupos do IAM e do Kubernetes necessárias para o perfil do IAM `my-console-viewer-role`, consulte Permissões obrigatórias.
- Uma seção `mapUsers` que mapeia o usuário do IAM `admin` da conta da AWS padrão para o grupo `system:masters` do Kubernetes e o usuário `my-user` de uma conta da AWS diferente mapeada para um grupo do Kubernetes que pode visualizar recursos do Kubernetes de um namespace específico. Para acessar uma lista das permissões dos grupos do IAM e do Kubernetesnecessárias para o usuário do IAM `my-user`, consulte Permissões obrigatórias.

Adicione ou remova linhas conforme a necessidade e substitua todos os *example values* pelos seus próprios valores.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
 file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::111122223333:role/my-role
      username: system:node:{{EC2PrivateDNSName}}
```

```
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
    - groups:
      - system:masters
      userarn: arn:aws:iam::111122223333:user/admin
      username: admin
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws:iam::444455556666:user/my-user
      username: my-user
```

3. Salve o arquivo e saia do seu editor de texto.

Como aplicar o **ConfigMapaws-auth** ao seu cluster

O `ConfigMap` `aws-auth` é criado e aplicado automaticamente ao cluster ao criar um grupo de nós gerenciados ou ao criar um grupo de nós usando `eksctl`. Ele é criado inicialmente para permitir que os nós ingressem no cluster, mas você também pode usar esse `ConfigMap` para adicionar acesso com controle de acesso baseado em função (RBAC) às entidades principais do IAM. Se já tiver iniciado os nós autogerenciados, mas ainda não tiver aplicado o `ConfigMap` `aws-auth` ao seu cluster, você poderá fazer isso usando o procedimento a seguir.

Para aplicar o **ConfigMapaws-auth** ao seu cluster

1. Verifique se você já aplicou o `ConfigMap` `aws-auth`.

   ```
   kubectl describe configmap -n kube-system aws-auth
   ```

   Se você receber um erro informando “`Error from server (NotFound): configmaps "aws-auth" not found`”, execute as etapas a seguir para aplicar o `ConfigMap` comum.

2. Faça download, edite e aplique o mapa de configuração do autenticador da AWS.

   a. Faça download do mapa de configuração.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-
   eks/cloudformation/2020-10-29/aws-auth-cm.yaml
   ```

b. No arquivo `aws-auth-cm.yaml`, defina o `rolearn` como o nome de recurso da Amazon (ARN) da função do IAM associada aos seus nós. É possível fazer isso com um editor de texto ou substituindo *`my-node-instance-role`* e executando o seguinte comando:

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
```

Não modifique outras linhas do arquivo.

> **Important**
>
> O ARN da função não pode incluir um caminho, como `role/my-team/developers/my-role`. O formato do ARN deve ser `arn:aws:iam::111122223333:role/my-role`. Neste exemplo, `my-team/developers/` precisa ser removido.

Você pode inspecionar as saídas de pilha do AWS CloudFormation para os grupos de nó e procurar os seguintes valores:

- InstanceRoleARN: para grupos de nós que foram criados com `eksctl`
- NodeInstanceRole: para grupos de nós que foram criados com modelos do AWS CloudFormation do Amazon EKS fornecidos no AWS Management Console

c. Aplique a configuração. Esse comando pode demorar alguns minutos para ser concluído.

```
kubectl apply -f aws-auth-cm.yaml
```

> **Note**
>
> Se você receber qualquer erro de autorização ou de tipo de recurso, consulte Acesso negado ou não autorizado (`kubectl`) no tópico de solução de problemas.

3. Observe o status de seus nós e aguarde até que eles atinjam o status `Ready`.

```
kubectl get nodes --watch
```

Insira `Ctrl+C` para retornar a um prompt de shell.

Conceda aos usuários acesso ao Kubernetes com um provedor OIDC externo

O Amazon EKS oferece suporte ao uso de provedores de identidade OpenID Connect (OIDC) como um método de autenticação de usuários no cluster. Os provedores de identidade OIDC podem ser usados com, ou como uma alternativa para, o AWS Identity and Access Management (IAM). Para obter mais informações sobre como usar a IAM, consulte the section called “Conceder acesso a APIs do Kubernetes”. Depois de configurar a autenticação do cluster, você pode criar `roles` e `clusterroles` do Kubernetes para atribuir permissões às funções e, em seguida, vincular as funções às identidades usando `rolebindings` e `clusterrolebindings` do Kubernetes. Para mais informações, consulte Using RBAC Authorization (Usar autorização RBAC) na documentação do Kubernetes.

Considerações

- Você pode associar um provedor de identidade OIDC ao cluster.
- O Kubernetes não fornece um provedor de identidade OIDC. Você pode usar um provedor de identidade OIDC público existente ou executar seu próprio provedor de identidade. Para uma lista de provedores certificados, consulte OpenID Certification (Certificação OpenID) no site OpenID.
- O URL do emissor do provedor de identidade OIDC deve ser acessível publicamente, para que o Amazon EKS possa descobrir as chaves de assinatura. O Amazon EKS não oferece suporte a provedores de identidade OIDC com certificados autoassinados.
- Não é possível desabilitar a autenticação IAM em seu cluster, porque ela ainda é necessária para juntar nós a um cluster.
- Um cluster do Amazon EKS ainda deve ser criado por uma entidade principal do IAM da AWS, em vez de um usuário do provedor de identidade OIDC. Isso ocorre porque o criador do cluster interage com as APIs do Amazon EKS, não com as APIs do Kubernetes.
- Os usuários autenticados pelo provedor de identidade OIDC serão listados no log de auditoria do cluster se os logs do CloudWatch estiverem ativados para o plano de controle. Para obter mais informações, consulte Habilitar ou desabilitar os logs do ambiente de gerenciamento.
- Não é possível fazer login no AWS Management Console com uma conta de um provedor OIDC. Você só pode visualizar os recursos do Kubernetes no console fazendo login no AWS Management Console com uma conta do AWS Identity and Access Management.

Associe um provedor de identidade OIDC

Antes de associar um provedor de identidade OIDC ao cluster, você precisa das seguintes informações do seu provedor:

URL do emissor

URL do provedor de identidade do OIDC que permite que o servidor de API descubra as chaves de assinatura públicas para verificar tokens. O URL deve começar com `https://` e deve corresponder à reivindicação `iss` nos tokens de ID do OIDC do provedor. De acordo com o padrão OIDC, os componentes de caminho são permitidos, mas os parâmetros de consulta não são. Normalmente, o URL consiste apenas em um nome de host, como `https://server.example.org` ou `https://example.com`. Este URL deve apontar para o nível abaixo de `.well-known/openid-configuration` e deve ser acessível ao público pela Internet.

ID do cliente (também conhecido como público)

O ID da aplicação do cliente que faz solicitações de autenticação ao provedor de identidades OIDC.

Você pode associar um provedor de identidade usando o `eksctl` ou o AWS Management Console.

eksctl

Para associar um provedor de identidade OIDC ao cluster usando o **`eksctl`**

1. Crie um arquivo denominado *`associate-identity-provider.yaml`* com o seguinte conteúdo: Substitua *`example values`* pelos seus próprios valores. Os valores na seção `identityProviders` são obtidos do seu provedor de identidade OIDC. Os valores são necessários apenas para as configurações `name`, `type`, `issuerUrl` e `clientId` em `identityProviders`.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: your-region-code
```

```
identityProviders:
  - name: my-provider
    type: oidc
    issuerUrl: https://example.com
    clientId: kubernetes
    usernameClaim: email
    usernamePrefix: my-username-prefix
    groupsClaim: my-claim
    groupsPrefix: my-groups-prefix
    requiredClaims:
      string: string
    tags:
      env: dev
```

> **Important**
>
> Não especifique o `system:`, ou qualquer parte dessa string, para `groupsPrefix` ou `usernamePrefix`.

2. Crie o provedor.

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. Para usar o `kubectl` para trabalhar com seu cluster e o provedor de identidade OIDC, consulte Como usar `kubectl` na documentação do Kubernetes.

AWS Management Console

Para associar um provedor de identidade OIDC ao cluster usando o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione seu cluster e, em seguida, selecione a guia Acesso.
3. Na seção Provedores de identidade OIDC, selecione Provedor de identidade do associado.
4. Na página Provedor de identidade OIDC do associado, insira ou selecione as opções a seguir e escolha Associar.
   - Em Name (Nome), insira um nome exclusivo para o provedor.
   - Para Issuer URL (URL do emissor), insira a URL do seu provedor. Este URL deve ser acessado pela Internet.

- Para o ID do cliente, insira o ID do cliente do provedor de identidade OIDC (também conhecido como público).
- Para o Username claim (Reivindicação do usuário), insira a reivindicação a ser usada como nome de usuário.
- Para o Username claim (Reivindicação do usuário), insira a reivindicação a ser usada como grupo de usuários.
- (Opcional) Selecione Advanced options (Opções avançadas) e insira ou selecione as informações a seguir.
  - Username prefix (Prefixo do usuário): insira um prefixo para preceder as reivindicações de nome de usuário. O prefixo precede as reivindicações de nome de usuário para evitar conflitos com nomes existentes. Se você não fornecer um valor e o nome de usuário for um valor diferente de `email`, o prefixo assume como padrão o valor para a URL do emissor. Você pode usar o valor `-` para desativar todos os prefixos. Não especifique o `system:`, ou qualquer parte dessa string.
  - Groups prefix (Prefixo de grupos): insira um prefixo para preceder as reivindicações de grupos. O prefixo precede as reivindicações de nome de usuário para evitar conflitos com nomes existentes (como `system: groups`). Por exemplo, o valor `oidc:` cria nomes de grupos como `oidc:engineering` e `oidc:infra`. Não especifique o `system:`, ou qualquer parte dessa string.
  - Required claims (Reivindicações necessárias): selecione Add claim (adicionar reivindicação) e insira um ou mais pares de valor de chave que descrevam as reivindicações necessárias no token de ID do cliente. Os pares descrevem as reivindicações obrigatórias no token de ID. Se definido, cada reivindicação é verificada para estar presente no token de ID com um valor correspondente.

5. Para usar o `kubectl` para trabalhar com seu cluster e o provedor de identidade OIDC, consulte Como usar `kubectl` na documentação do Kubernetes.

Exemplo de política do IAM

Se você quiser impedir que um provedor de identidade OIDC seja associado a um cluster, crie e associe a política do IAM a seguir às contas do IAM dos seus administradores do Amazon EKS. Para obter mais informações, consulte Creating IAM policies (Criar políticas do IAM) e Adding IAM identity permissions (Adicionar permissões de identidade do IAM) no Manual do usuário do IAM e também Actions, resources, and condition keys for Amazon Elastic Kubernetes Service (Ações, recursos e

chaves de condição para o Amazon Elastic Kubernetes Service) na Referência da autorização de serviço.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "denyOIDC",
            "Effect": "Deny",
            "Action": [
                "eks:AssociateIdentityProviderConfig"
            ],
            "Resource": "arn:aws:eks:us-west-2.amazonaws.com:111122223333:cluster/*"

        },
        {
            "Sid": "eksAdmin",
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

A política de exemplo a seguir permite a associação do provedor de identidade OIDC se o `clientID` for `kubernetes` e `issuerUrl` for `https://cognito-idp.us-west-2amazonaws.com/*`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCognitoOnly",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotLikeIfExists": {
                    "eks:issuerUrl": "https://cognito-idp.us-west-2.amazonaws.com/*"
                }
```

```
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
                    "eks:clientId": "kubernetes"
                }
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}
```

Provedores de identidade OIDC validados por parceiros

O Amazon EKS mantém um relacionamento com uma rede de parceiros que oferecem suporte para provedores compatíveis de identidade OIDC. Consulte a documentação dos parceiros a seguir para obter detalhes sobre como integrar o provedor de identidade ao Amazon EKS.

Parceiro	Produto	Documentação
PingIdentity	PingOne for Enterprise	Instruções de instalação

O Amazon EKS tem como objetivo oferecer uma ampla variedade de opções para cobrir todos os casos de uso. Se você desenvolver um provedor de identidade compatível com OIDC e comercialmente disponível que não esteja listado aqui, entre em contato com nossa equipe de parceiros em aws-container-partners@amazon.com para obter mais informações.

Desassociar um provedor de identidade OIDC do cluster

Se você desassociar um provedor de identidade OIDC do cluster, os usuários incluídos no provedor não poderão mais acessar o cluster. No entanto, você ainda pode acessar o cluster com as entidades principais do IAM.

Para desassociar um provedor de identidade OIDC do cluster usando o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Na seção Provedores de identidade OIDC, selecione Desassociar, insira o nome do provedor de identidade e selecione `Disassociate`.

Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig

Neste tópico, você cria um arquivo `kubeconfig` para o cluster (ou atualiza um existente).

A ferramenta da linha de comando `kubectl` usa informações de configuração em arquivos `kubeconfig` para se comunicar com o servidor de API de um cluster. Para obter mais informações, consulte Organizando o acesso ao cluster usando arquivos kubeconfig na documentação do Kubernetes.

O Amazon EKS usa o comando `aws eks get-token` com `kubectl` para autenticação de cluster. Por padrão, a AWS CLI usa as mesmas credenciais que são retornadas com o seguinte comando:

```
aws sts get-caller-identity
```

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.
- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para

verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.

- Um usuário ou perfil do IAM com permissão para usar a ação de API `eks:DescribeCluster` para o cluster que você especificar. Para ter mais informações, consulte Exemplos de políticas baseadas em identidade do Amazon EKS. Se você usa uma identidade do seu próprio provedor OpenID Connect para acessar seu cluster, consulte Como usar `kubectl` na documentação do Kubernetes para criar ou atualizar o arquivo `kube config`.

Criar arquivo **`kubeconfig`** automaticamente

Pré-requisitos

- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.
- Permissão para usar a ação de API `eks:DescribeCluster` para o cluster que você especificar. Para ter mais informações, consulte Exemplos de políticas baseadas em identidade do Amazon EKS.

Para criar o arquivo **`kubeconfig`** com a AWS CLI

1. Crie um arquivo `kubeconfig` para o cluster. Substitua *`region-code`* pela Região da AWS em que seu cluster está localizado e substitua *`my-cluster`* pelo nome do seu cluster.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Por padrão, o arquivo de configuração resultante é criado no caminho padrão do `kubeconfig` (`.kube`) em seu diretório inicial ou mesclado com um `config` existente no local. Você pode especificar outro caminho com a opção **`--kubeconfig`**.

Você pode especificar um ARN de função do IAM com a opção **`--role-arn`** para uso na autenticação quando você emitir comandos `kubectl`. Caso contrário, a entidade principal do IAM na AWS CLI padrão ou na cadeia de credenciais padrão do SDK será usada. Você pode visualizar sua identidade padrão da AWS CLI ou do SDK executando o comando `aws sts get-caller-identity`.

Para todas as opções disponíveis, execute o comando `aws eks update-kubeconfig help` ou consulte `update-kubeconfig` na Referência de comando da AWS CLI.

2. Teste a configuração.

```
kubectl get svc
```

Veja um exemplo de saída abaixo.

```
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

Se você receber qualquer erro de autorização ou de tipo de recurso, consulte Acesso negado ou não autorizado (`kubectl`) no tópico de solução de problemas.

Conceder às workloads do Kubernetes acesso à AWS usando contas de serviço da Kubernetes

Uma conta de serviço do Kubernetes fornece uma identidade para processos executados em um Pod. Para obter mais informações, consulte Managing Service Accounts (Gerenciar contas de serviço) na documentação do Kubernetes. Se o Pod precisar de acesso a serviços da AWS, você poderá mapear a conta de serviço para uma identidade do AWS Identity and Access Management para conceder esse acesso. Para ter mais informações, consulte Perfis do IAM para contas de serviço.

Tokens de contas de serviço

O recurso `BoundServiceAccountTokenVolume` está habilitado por padrão nas versões Kubernetes. Esse recurso melhora a segurança de tokens de contas de serviços, permitindo que workloads sejam executadas no Kubernetes para solicitar tokens de Web JSON que são restritos a um público, prazo e chave.. Tokens de contas de serviço têm uma expiração de uma hora. Em versões anteriores do Kubernetes, os tokens não tinham expiração. Isso significa que os clientes que dependem desses tokens precisam atualizá-los em até uma hora. Os exemplos a seguir de SDKs de cliente do Kubernetes atualizam os tokens automaticamente dentro do prazo exigido:

- Versão do Go `0.15.7` e posteriores
- Versão do Python `12.0.0` e posteriores
- Versão Java `9.0.0` e posterior
- Versão JavaScript `0.10.3` e posterior
- Ramificação Ruby `master`
- Versão do Haskell `0.3.0.0`
- Versão do C# `7.0.5` ou posterior.

Se sua workload estiver usando uma versão anterior do cliente, você deverá atualizá-la. Para permitir uma migração suave de clientes para os tokens de conta de serviço com limite de tempo mais recentes, o Kubernetes adiciona um período de expiração estendido ao token da conta de serviço além do tempo padrão de uma hora. Para clusters do Amazon EKS, o período de expiração prolongado equivale a 90 dias. O servidor de API do Kubernetes do seu cluster do Amazon EKS rejeita solicitações com tokens com mais de 90 dias. É recomendável verificar as aplicações e suas dependências para garantir que os SDKs de cliente do Kubernetes sejam iguais ou posteriores às versões listadas anteriormente.

Quando o servidor de API recebe solicitações com tokens com mais de uma hora, ele anota o evento de logs de auditoria da API com `annotations.authentication.k8s.io/stale-token`. O valor da anotação é parecido com o exemplo a seguir:

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold:
 4185802.
```

Se o cluster tiver registro em log do ambiente de gerenciamento habilitado, as anotações estarão nos logs de auditoria. É possível utilizar a seguinte consulta do Cloudwatch Log Insights para identificar todos os Pods no cluster do Amazon EKS que estejam utilizando tokens obsoletos:

```
fields @timestamp
| filter @logStream like /kube-apiserver-audit/
| filter @message like /seconds after warning threshold/
| parse @message "subject: *, seconds after warning threshold:*\"" as subject,
 elapsedtime
```

O `subject` refere-se à conta de serviço que o Pod utilizou. O `elapsedtime` indica o tempo decorrido (em segundos) após a leitura do token mais recente. As solicitações para o servidor de API são negadas quando o `elapsedtime` excede 90 dias (7.776.000 segundos). É necessário atualizar proativamente o SDK do cliente do Kubernetes das suas aplicações para utilizar uma das versões listadas anteriormente que atualizam o token automaticamente. Se o token da conta de serviço utilizado estiver se aproximando dos 90 dias e não houver tempo suficiente para atualizar as versões do SDK de cliente antes da expiração desse token, você pode encerrar os Pods existentes e criar novos. Isso provoca a reformulação do token da conta de serviço, fornecendo mais 90 dias para você atualizar os SDKs da versão do cliente.

Se o Pods fizer parte de uma implantação, a maneira sugerida de encerrar Pod e, ao mesmo tempo, manter a alta disponibilidade é executar uma implantação com o comando a seguir. Substitua *`my-deployment`* pelo nome da implantação.

```
kubectl rollout restart deployment/my-deployment
```

Complementos de cluster

Os seguintes complementos de cluster foram atualizados para utilizar os SDKs de cliente Kubernetes, que redefinem tokens de conta de serviço automaticamente: Convém garantir que as versões listadas ou posteriores estejam instaladas no seu cluster.

- Amazon VPC CNI plugin for Kubernetes e versões de plug-ins auxiliares de métricas `1.8.0` e posteriores. Para verificar sua versão atual ou atualizá-la, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS e cni-metrics-helper.
- CoreDNS versão `1.8.4` e versões posteriores. Para conferir sua versão atual ou atualizá-la, consulte Trabalhando com o complemento CoreDNS do Amazon EKS.

- AWS Load Balancer Controller versão `2.0.0` e versões posteriores. Para conferir sua versão atual ou atualizá-la, consulte O que é o AWS Load Balancer Controller?.
- Uma Versão atual do `kube-proxy`. Para conferir sua versão atual ou atualizá-la, consulte Trabalhando com o complemento `kube-proxy` do Kubernetes.
- AWS para a versão Fluent Bit `2.25.0` ou posterior. Para atualizar a versão atual, consulte Releases (Versões), no GitHub.
- Versão de imagem do Fluentd 1.14.6-1.2 ou posterior e plugin de filtro do Fluentd para metadados do Kubernetes versão 2.11.1 ou posterior.

Conceder permissões do AWS Identity and Access Management para workloads em clusters do Amazon Elastic Kubernetes Service

O Amazon EKS fornece duas maneiras de conceder permissões do AWS Identity and Access Management para workloads executadas em clusters do Amazon EKS: perfis do IAM para contas de serviço e EKS Pods Identities.

Perfis do IAM para contas de serviço

> Os perfis do IAM para contas de serviço (IRSA) configuram aplicações Kubernetes em execução na AWS com permissões minuciosas do IAM para acessar vários outros recursos da AWS, como buckets do Amazon S3, tabelas do Amazon DynamoDB e muito mais. É possível executar várias aplicações juntas no mesmo cluster do Amazon EKS e garantir que cada aplicação tenha apenas o conjunto mínimo de permissões de que precisa. O IRSA foi criado para oferecer suporte a várias opções de implantação do Kubernetes aceitas pela AWS, como Amazon EKS, Amazon EKS Anywhere, Serviço Red Hat OpenShift na AWS e clusters do Kubernetes autogerenciados em instâncias do Amazon EC2. Assim, o IRSA foi criado usando um serviço da AWS básico, como o IAM, e não dependia diretamente do serviço Amazon EKS e da API do EKS. Para ter mais informações, consulte Perfis do IAM para contas de serviço.

EKS Pod Identities

> O EKS Pod Identity oferece aos administradores de clusters um fluxo de trabalho simplificado para autenticar aplicações para acessar vários outros recursos da AWS, como buckets do Amazon S3, tabelas do Amazon DynamoDB e muito mais. O EKS Pod Identity é somente para EKS e, como resultado, simplifica como os administradores de cluster podem configurar aplicações Kubernetes para obter permissões do IAM. Agora, essas permissões podem ser configuradas com facilidade em menos etapas diretamente via AWS Management Console,

API do EKS e AWS CLI, e não há nenhuma ação a ser executada dentro do cluster em nenhum objeto Kubernetes. Os administradores de cluster não precisam alternar entre os serviços EKS e IAM nem usar operações privilegiadas do IAM para configurar as permissões exigidas por suas aplicações. Os perfis do IAM agora podem ser usados em vários clusters sem a necessidade de atualizar a política de confiança do perfil ao criar novos clusters. As credenciais do IAM fornecidas pelo EKS Pod Identity incluem tags de sessão de perfil, com atributos como nome do cluster, namespace e nome da conta de serviço. As tags de sessão de perfil permitem que os administradores criem um único perfil que pode funcionar em várias contas de serviço, permitindo o acesso aos recursos da AWS com base em tags correspondentes. Para ter mais informações, consulte Saiba como a EKS Pod Identity concede aos pods acesso aos serviços da AWS.

Comparação entre o EKS Pod Identity e o IRSA

Em alto nível, tanto o EKS Pod Identity quanto o IRSA permitem conceder permissões do IAM para aplicações executadas em clusters do Kubernetes. No entanto, eles são fundamentalmente diferentes na forma como você os configura, nos limites aceitos e nos recursos ativados. Abaixo, comparamos algumas das principais características de ambas as soluções.

	EKS Pod Identity	IRSA
Extensibilidade de perfis	É necessário configurar cada perfil uma vez para estabelecer confiança com a recém-introduzida entidade principal do serviço Amazon EKS `pods.eks.amazonaws.com` . Após essa etapa única, não será necessário atualizar a política de confiança do perfil sempre que ela for usada em um novo cluster.	Você precisará atualizar a política de confiança do perfil do IAM com o novo endpoint do provedor OIDC do cluster EKS sempre que quiser usar o perfil em um novo cluster.
Escalabilidade de clusters	O EKS Pod Identity não exige que os usuários configurem o provedor OIDC do IAM,	Cada cluster do EKS tem um URL do emissor OpenID Connect (OIDC) associado a ele. Para usar o IRSA, é

	EKS Pod Identity	IRSA
	portanto, esse limite não se aplica.	necessário criar um provedor OpenID Connect exclusivo para cada cluster do EKS no IAM. O IAM tem um limite global padrão de 100 provedores OIDC para cada Conta da AWS. Se pretende ter mais de 100 clusters do EKS para cada Conta da AWS com IRSA, você atingirá o limite de OIDC provedores do IAM.
Escalabilidade de funções	O EKS Pod Identity não exige que os usuários definam uma relação de confiança entre o perfil do IAM e a conta de serviço na política de confiança, portanto, esse limite não se aplica.	No IRSA, é necessário definir a relação de confiança entre um perfil do IAM e uma conta de serviço na política de confiança do perfil. Por padrão, a duração do tamanho da política de confiança é `2048`. Isso significa que você pode definir quatro relações de confiança normalmen te em uma única política de confiança. Embora seja possível aumentar o limite de duração da política de confiança, em geral você tem um limite máximo de 8 relações de confiança em uma única política de confiança.

	EKS Pod Identity	IRSA
Reutilização de perfis	As credenciais temporárias do AWS STS fornecidas pelo EKS Pod Identity incluem tags de sessão de perfil, com atributos como nome do cluster, namespace e nome da conta de serviço. As tags de sessão de perfil permitem que os administradores criem um único perfil do IAM que pode ser usado com várias contas de serviço, com diferentes permissões efetivas, permitind o acesso a recursos da AWS com base nas tags anexadas a elas. Isso também é chamado controle de acesso por atributo (ABAC). Para ter mais informações, consulte Conceder acesso aos pods a recursos da AWS baseados em tags.	Não há suporte a tags de sessão AWS STS. Você pode reutilizar um perfil entre clusters, mas cada pod recebe todas as permissões do perfil.
Ambientes compatíveis	O EKS Pod Identity só está disponível no Amazon EKS.	O IRSA pode ser usado como Amazon EKS, Amazon EKS Anywhere, Serviço Red Hat OpenShift na AWS e clusters do Kubernetes autogeren ciados em instâncias do Amazon EC2.

	EKS Pod Identity	IRSA
Versões do EKS compatíveis	EKS Kubernetes versões `1.24` ou posteriores. Para obter números de versão específicos, consulte Versões de cluster do EKS Pod Identity.	Todas as versões de cluster do EKS compatíveis.

Saiba como a EKS Pod Identity concede aos pods acesso aos serviços da AWS

Aplicações nos contêineres de um Pod podem usar um AWS SDK ou AWS CLI para fazer solicitações de API aos Serviços da AWS usando permissões do AWS Identity and Access Management (IAM). As aplicações devem assinar suas solicitações de API da AWS com as credenciais da AWS.

Os EKS Pod Identities permitem gerenciar credenciais para suas aplicações de forma semelhante a como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2. Em vez de criar e distribuir suas credenciais da AWS aos contêineres ou de usar o perfil da instância do Amazon EC2, você pode associar um perfil do IAM a uma conta de serviço do Kubernetes e configurar os Pods para usar a conta de serviço.

Cada associação ao EKS Pod Identity mapeia um perfil a uma conta de serviço em um namespace no cluster especificado. Se você tiver a mesma aplicação em vários clusters, poderá fazer associações idênticas em cada cluster sem modificar a política de confiança do perfil.

Se um pod usa uma conta de serviço que tem uma associação, o Amazon EKS define variáveis de ambiente nos contêineres do pod. As variáveis de ambiente configuram os AWS SDKs, incluindo o AWS CLI, para usar as credenciais do EKS Pod Identity.

Benefícios dos EKS Pod Identities

Os EKS Pod Identities fornecem os seguintes benefícios:

- Menor privilégio: é possível definir o escopo de permissões do IAM para uma conta de serviço, e somente os Pods que usarem essa conta de serviço terão acesso a essas permissões. Esse recurso também elimina a necessidade de soluções de terceiros, como o `kiam` ou o `kube2iam`.

- Isolamento de credenciais: um contêiner de Pod's só pode recuperar credenciais para o perfil do IAM que esteja associada à conta de serviço que o contêiner use. Um contêiner nunca tem acesso a credenciais usadas por outros contêineres em outros Pods. Ao usar identidades de Pods, os contêineres de Pod's também têm as permissões atribuídas ao perfil do IAM do nó do Amazon EKS, a menos que você bloqueie o acesso do Pod ao serviço de metadados de instância (IMDS) do Amazon EC2. Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.
- Auditabilidade:: o acesso e os logs de eventos estão disponíveis por meio do AWS CloudTrail para ajudar a facilitar a auditoria retrospectiva.

O EKS Pod Identity é um método mais simples do que Perfis do IAM para contas de serviço, pois esse método não usa provedores de identidade OIDC. O EKS Pod Identity oferece os seguintes aprimoramentos:

- Operações independentes: em muitas organizações, criar provedores de identidade OIDC é responsabilidade de equipes diferentes da administração dos clusters do Kubernetes. O EKS Pod Identity tem uma separação clara dos deveres em que toda a configuração das associações ao EKS Pod Identity é feita no Amazon EKS e toda a configuração das permissões do IAM é feita no IAM.
- Reutilização: o EKS Pod Identity usa uma única entidade principal do IAM em vez das entidades principais separadas para cada cluster que os perfis do IAM para contas de serviço utilizam. Seu administrador do IAM adiciona a seguinte entidade principal à política de confiança de qualquer perfil para torná-la utilizável pelo EKS Pod Identities.

```
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            }
```

- Escalabilidade: cada conjunto de credenciais temporárias é assumido pelo serviço EKS Auth no EKS Pod Identity, em vez de cada AWS SDK que você executa em cada pod. Em seguida, o Amazon EKS Pod Identity Agent executado em cada nó emite as credenciais para os SDKs. Assim, a carga é reduzida para uma vez para cada nó e não é duplicada em cada pod. Para obter mais detalhes do processo, consulte Entender como a EKS Pod Identity funciona.

Para obter mais informações sobre como comparar as duas alternativas, consulte Conceder às workloads do Kubernetes acesso à AWS usando contas de serviço da Kubernetes.

Visão geral da configuração do EKS Pod Identities

Ative o EKS Pod Identities concluindo os seguintes procedimentos:

1. Configurar o Amazon EKS Pod Identity Agent: você só conclui este procedimento uma vez para cada cluster.
2. Atribuir um perfil do IAM a uma conta de serviço do Kubernetes: conclua este procedimento para cada conjunto exclusivo de permissões que você deseja que uma aplicação tenha.
3. Configurar pods para acessar serviços da AWS com contas de serviço: conclua este procedimento para cada Pod que precisa acessar Serviços da AWS.
4. Usar identidade de pods com o AWS SDK: confirme se a workload usa um AWS SDK de uma versão compatível e se a workload usa a cadeia de credenciais padrão.

Considerações sobre o EKS Pod Identity

- É possível associar um perfil do IAM a cada conta de serviço do Kubernetes em cada cluster. É possível alterar qual perfil é mapeado na conta de serviço editando a associação ao EKS Pod Identity.
- Só é possível associar perfis que estejam na mesma Conta da AWS que o cluster. É possível delegar o acesso de outra conta ao perfil nessa conta configurada por você para o EKS Pod Identities usar. Para ver um tutorial sobre como delegar acesso e `AssumeRole`, consulte o Delegar acesso em contas da AWS usando perfis do IAM no Guia do usuário do IAM.
- O EKS Pod Identity Agent é necessário. Ele é executado como um `DaemonSet` do Kubernetes em seus nós e fornece credenciais somente para os pods no nó em que for executado. Para obter mais informações sobre a compatibilidade do EKS Pod Identity Agent, consulte a seção Restrições do EKS Pod Identity a seguir.
- O EKS Pod Identity Agent usa o `hostNetwork` do nó e usa a porta `80` e a porta `2703` em um endereço link-local no nó. Esse endereço é `169.254.170.23` para clusters IPv4 e `[fd00:ec2::23]` para clusters IPv6.

  Se você desabilitar endereços `IPv6` ou de alguma forma impedir endereços IP `IPv6` do localhost, o agente não poderá iniciar. Para iniciar o agente em nós que não podem usar `IPv6`, siga as etapas em Desabilitar o `IPv6` no EKS Pod Identity Agent para desabilitar a configuração de `IPv6`.

Versões de cluster do EKS Pod Identity

Para usar o EKS Pod Identities, o cluster deve ter uma versão da plataforma igual ou posterior à versão listada na tabela a seguir ou uma versão de Kubernetes posterior às versões listadas na tabela.

Versão do Kubernetes	Versão da plataforma
`1.30`	`eks.2`
`1.29`	`eks.1`
`1.28`	`eks.4`
`1.27`	`eks.8`
`1.26`	`eks.9`
`1.25`	`eks.10`
`1.24`	`eks.13`

Versões de complementos compatíveis com o EKS Pod Identity

> **Important**
>
> Para usar o EKS Pod Identity com um complemento do EKS, é necessário criar a associação ao EKS Pod Identity manualmente. Não escolha um perfil do IAM na configuração do complemento no AWS Management Console, esse perfil só é usado com o IRSA.

Os complementos e complementos autogerenciados do Amazon EKS que precisam de credenciais do IAM podem usar o EKS Pod Identity, o IRSA ou o perfil da instância. A lista de complementos que usam credenciais do IAM e oferecem suporte ao EKS Pod Identity é:

- Amazon VPC CNI plugin for Kubernetes `1.15.5-eksbuild.1` ou posterior
- AWS Load Balancer Controller `2.7.0` ou posterior. Observe que o AWS Load Balancer Controller não está disponível como complemento do EKS, mas está disponível como um complemento autogerenciado.

Restrições do EKS Pod Identity

Os EKS Pod Identities estão disponíveis em:

- Versões de cluster do Amazon EKS listadas no tópico Versões de cluster do EKS Pod Identity anterior.
- Nós de processamento no cluster que são instâncias do Linux do Amazon EC2.

Os EKS Pod Identities não estão disponíveis em:

- AWS GovCloud (US).
- AWS Outposts.
- Amazon EKS Anywhere
- Clusters do Kubernetes que criados e executados no Amazon EC2. Os componentes do EKS Pod Identity estão disponíveis somente no Amazon EKS.

Não é possível usar o EKS Pod Identities com:

- Pods executados em qualquer lugar, exceto instâncias do Linux do Amazon EC2. Não há suporte a pods Linux e Windows executados no AWS Fargate (Fargate). Não há suporte a pods executados em instâncias do Windows do Amazon EC2.
- Complementos do Amazon EKS que precisam de credenciais do IAM. Em vez disso, os complementos do EKS só podem usar perfis do IAM para contas de serviço. A lista de complementos do EKS que usam credenciais do IAM inclui:
  - Os drivers de armazenamento do CSI: CSI do EBS, CSI do EFS, driver da CSI do Amazon FSx para Lustre, driver da CSI do Amazon FSx para NetApp ONTAP, driver da CSI do Amazon FSx para OpenZFS, driver da CSI do Amazon File Cache, AWS Secrets and Configuration Provider (ASCP) para o driver da CSI do Kubernetes Secrets Store

> **Note**
>
> Se esses controladores, drivers e plug-ins forem instalados como complementos autogerenciados em vez de complementos do EKS, eles serão compatíveis com o EKS Pod Identities, desde que sejam atualizados para usar os SDKs da AWS mais recentes.

Entender como a EKS Pod Identity funciona

As associações do Amazon EKS Pod Identity permitem gerenciar credenciais para suas aplicações de forma semelhante a como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2.

O Amazon EKS Pod Identity fornece credenciais para suas workloads com uma API EKS Auth adicional e um pod de agente que é executado em cada nó.

Em seus complementos, como os complementos do Amazon EKS e controladores autogerenciados, operadores e outros complementos, o autor precisa atualizar o software para usar os AWS SDKs mais recentes. Para ver a lista de compatibilidade entre o EKS Pod Identity e os complementos produzidos pelo Amazon EKS, consulte a seção Restrições do EKS Pod Identity anterior.

Usar o EKS Pod Identities em seu código

É possível usar os AWS SDKs no código para acessar os serviços da AWS. Escreva código para criar um cliente para um serviço da AWS com um SDK e, por padrão, o SDK pesquisa em uma cadeia de locais as credenciais do AWS Identity and Access Management que serão usadas. Depois que as credenciais válidas forem encontradas, a pesquisa será interrompida. Para obter mais informações sobre os locais padrão usados, consulte a Cadeia de provedores de credenciais no Guia de referência de SDKs e ferramentas da AWS.

As identidades do EKS Pod foram adicionadas ao Provedor de credenciais do container, o qual é pesquisado em uma etapa na cadeia de credenciais padrão. Se atualmente suas workloads usam credenciais mais antigas na cadeia de credenciais, elas continuarão sendo usadas mesmo se você configurar uma associação do EKS Pod Identity para a mesma workload. Dessa forma, você pode migrar com segurança de outros tipos de credenciais criando a associação antes de remover as credenciais antigas.

O provedor de credenciais do contêiner fornece credenciais temporárias de um agente executado em cada nó. No Amazon EKS, o agente é o Amazon EKS Pod Identity Agent e, no Amazon Elastic Container Service, o agente é o `amazon-ecs-agent`. Os SDKs usam variáveis de ambiente para localizar os agentes ao qual se conectarão.

Por outro lado, os perfis do IAM para contas de serviço fornecem um token de identidade da Web que o AWS SDK deve trocar com o AWS Security Token Service usando `AssumeRoleWithWebIdentity`.

Como o EKS Pod Identity Agent funciona com um Pod

1. Quando o Amazon EKS inicia um novo pod que usa uma conta de serviço com uma associação ao EKS Pod Identity, o cluster adiciona o seguinte conteúdo ao manifesto Pod:

```
    env:
    - name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
      value: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-
identity-token"
    - name: AWS_CONTAINER_CREDENTIALS_FULL_URI
      value: "http://169.254.170.23/v1/credentials"
    volumeMounts:
    - mountPath: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/"
      name: eks-pod-identity-token
  volumes:
  - name: eks-pod-identity-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          audience: pods.eks.amazonaws.com
          expirationSeconds: 86400 # 24 hours
          path: eks-pod-identity-token
```

2. Kubernetes seleciona em qual nó o pod será executado. Em seguida, o Amazon EKS Pod Identity Agent no nó usa a ação AssumeRoleForPodIdentity para recuperar credenciais temporárias da API de autenticação do EKS.
3. O EKS Pod Identity Agent disponibiliza essas credenciais para os AWS SDKs que você executa em seus contêineres.
4. Você usa o SDK em sua aplicação sem especificar um provedor de credenciais para usar a cadeia de credenciais padrão. Ou você especifica o provedor de credenciais do contêiner. Para obter mais informações sobre os locais padrão usados, consulte a Cadeia de provedores de credenciais no Guia de referência de SDKs e ferramentas da AWS.
5. O SDK usa as variáveis de ambiente para se conectar ao EKS Pod Identity Agent e recuperar as credenciais.

> **Note**
>
> Se suas workloads usam no momento credenciais mais antigas na cadeia de credenciais, essas credenciais continuarão sendo usadas mesmo se você configurar uma associação do EKS Pod Identity para a mesma workload.

Configurar o Amazon EKS Pod Identity Agent

As associações do Amazon EKS Pod Identity permitem gerenciar credenciais para suas aplicações de forma semelhante a como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2.

O Amazon EKS Pod Identity fornece credenciais para suas workloads com uma API EKS Auth adicional e um pod de agente executado em cada nó.

Considerações

- **`IPv6`**

  Por padrão, o EKS Pod Identity Agent escuta em um endereço `IPv4` e `IPv6` para os pods para solicitar credenciais. O agente usa o endereço IP de loopback (localhost) `169.254.170.23` para `IPv4` e o endereço IP de localhost `[fd00:ec2::23]` para `IPv6`.

  Se você desabilitar endereços `IPv6` ou de alguma forma impedir endereços IP `IPv6` do localhost, o agente não poderá iniciar. Para iniciar o agente em nós que não podem usar `IPv6`, siga as etapas em Desabilitar o `IPv6` no EKS Pod Identity Agent para desabilitar a configuração de `IPv6`.

Criar o Amazon EKS Pod Identity Agent

Pré-requisitos do agente

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS. A versão do cluster e a versão da plataforma devem ser iguais ou posteriores às versões listadas em Versões de cluster do EKS Pod Identity.
- O perfil de nó tem permissões para o agente realizar a ação `AssumeRoleForPodIdentity` na API de autenticação do EKS. É possível usar Política gerenciada da AWS: AmazonEKSWorkerNodePolicy ou adicionar uma política personalizada semelhante a esta:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity"
            ],
            "Resource": "*"
        }
    ]
}
```

Esta ação pode ser limitada por tags para restringir quais funções podem ser assumidas pelos pods que usam o agente.

- Os nós podem acessar e baixar imagens do Amazon ECR. A imagem do contêiner do complemento está nos registros listados em Visualizar registros de imagens de contêineres da Amazon para complementos do Amazon EKS.

  Observe que é possível alterar a localização da imagem e fornecer `imagePullSecrets` para complementos do EKS nas Configurações opcionais no AWS Management Console, em `--configuration-values` e na AWS CLI.
- Os nós podem acessar a API de autenticação do Amazon EKS. Para clusters privados, o endpoint `eks-auth` em AWS PrivateLink é obrigatório.

AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação esquerdo, selecione Clusters e depois o nome do cluster para o qual você deseja configurar o complemento do EKS Pod Identity Agent.
3. Escolha a guia Add-ons (Complementos).
4. Escolha Obter mais complementos.
5. Selecione a caixa no canto superior direito da caixa do complemento do EKS Pod Identity Agent e escolha Editar.
6. Na página Configurar complementos selecionados, selecione qualquer versão na lista suspensa Versão.

7. (Opcional) Expanda Configurações opcionais para inserir configurações adicionais. Por exemplo, é possível indicar um local alternativo para a imagem do contêiner e `ImagePullSecrets`. O JSON Schema com chaves aceitas é mostrado no Esquema de configuração do complemento.

   Insira as chaves e os valores de configuração em Valores de configuração.
8. Escolha Próximo.
9. Confirme que os pods do EKS Pod Identity Agent estão em execução no cluster.

   ```
   kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
   ```

   Veja um exemplo de saída abaixo.

   ```
   eks-pod-identity-agent-gmqp7                                          1/1
    Running   1 (24h ago)   24h
   eks-pod-identity-agent-prnsh                                          1/1
    Running   1 (24h ago)   24h
   ```

   Agora é possível usar associações do EKS Pod Identity em seu cluster. Para ter mais informações, consulte Atribuir um perfil do IAM a uma conta de serviço do Kubernetes.

AWS CLI

1. Execute o seguinte comando AWS CLI. Substitua o `my-cluster` pelo nome do cluster.

   ```
   aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-
   agent --addon-version v1.0.0-eksbuild.1
   ```

   > **Note**
   >
   > O EKS Pod Identity Agent não usa o `service-account-role-arn` para perfis do IAM para contas de serviço. Você deve fornecer ao EKS Pod Identity Agent as permissões no perfil de nó.

2. Confirme que os pods do EKS Pod Identity Agent estão em execução no cluster.

   ```
   kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
   ```

Veja um exemplo de saída abaixo.

```
eks-pod-identity-agent-gmqp7                                          1/1
 Running   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1
 Running   1 (24h ago)   24h
```

Agora é possível usar associações do EKS Pod Identity em seu cluster. Para ter mais informações, consulte Atribuir um perfil do IAM a uma conta de serviço do Kubernetes.

Atribuir um perfil do IAM a uma conta de serviço do Kubernetes

Este tópico aborda como configurar uma conta de serviço do Kubernetes para assumir um perfil do AWS Identity and Access Management (IAM) com o EKS Pod Identity. Todos os Pods configurados para usar a conta de serviço podem então acessar quaisquer Serviço da AWS para os quais a função tenha permissões de acesso.

Para criar uma associação ao EKS Pod Identity, há apenas uma única etapa: crie a associação no EKS via AWS Management Console, AWS CLI, AWS SDKs, AWS CloudFormation e outras ferramentas. Não há dados ou metadados sobre as associações dentro do cluster em nenhum objeto do Kubernetes, e você não adiciona anotações às contas de serviço.

Pré-requisitos

- Um cluster existente. Se você não tiver, poderá criar um, seguindo um dos guias Começar a usar o Amazon EKS.
- A entidade principal do IAM que está criando a associação deve ter o `iam:PassRole`.
- A versão mais recente do AWS CLI instalada e configurada em seu dispositivo ou no AWS CloudShell. É possível verificar sua versão atual com `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o `aws configure` no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório base no Guia do usuário do AWS CloudShell.

- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.
- Um arquivo `kubectl config` existente que contém a configuração do seu cluster. Para criar um arquivo `kubectl config`, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig.

Criar uma associação ao EKS Pod Identity

AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação esquerdo, selecione Clusters e depois o nome do cluster para o qual você deseja configurar o complemento do EKS Pod Identity Agent.
3. Escolha a guia Acesso.
4. Nas Associações do Pod Identity, escolha Criar.
5. Para o Perfil do IAM, selecione o perfil do IAM com as permissões que você deseja que a workload tenha.

> **Note**
>
> A lista contém apenas perfis que têm a política de confiança a seguir que permite que o EKS Pod Identity as use.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
```

```
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`

O EKS Pod Identity usa `AssumeRole` para assumir o perfil do IAM antes de passar as credenciais temporárias para seus pods.

`sts:TagSession`

O EKS Pod Identity usa `TagSession` para incluir tags de sessão nas solicitações para AWS STS.

Você pode usar essas tags nas condition keys na política de confiança para restringir quais contas de serviço, namespaces e clusters podem usar esse perfil.

Para obter uma lista de chaves de condição do Amazon EKS, consulte Condições definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar a chave de condição, consulte Ações definidas pelo Amazon Elastic Kubernetes Service.

6. Para o namespace Kubernetes, selecione o namespace do Kubernetes que contém a conta de serviço e a workload. Opcionalmente, é possível especificar um namespace por nome que não existe no cluster.
7. Para a conta de serviço do Kubernetes, selecione a conta de serviço do Kubernetes a ser usada. O manifesto da workload do Kubernetes deve especificar essa conta de serviço. Opcionalmente, é possível especificar uma conta de serviço por nome que não existe no cluster.
8. (Opcional) Para as Tags, escolha Adicionar tag para adicionar metadados em um par de chave e valor. Essas tags são aplicadas à associação e podem ser usadas nas políticas do IAM.

   Repita essa etapa para adicionar várias tags.
9. Escolha Criar.

AWS CLI

1. Se você quiser associar uma política do IAM existente ao seu perfil do IAM, vá para a próxima etapa.

   Crie uma política do IAM. É possível criar a sua própria política ou copiar uma política gerenciada da AWS que já conceda algumas das permissões de que você precisa e a personalizar de acordo com seus requisitos específicos. Para obter mais informações, consulte Criar políticas do IAM no Guia do usuário do IAM.

   a. Crie um arquivo que inclua as permissões para os Serviços da AWS que você deseja que seus Pods acessem. Para obter uma lista de todas as ações para todos os Serviços da AWS, consulte a Referência de autorização do serviço.

      É possível executar o comando a seguir para criar um arquivo de política de exemplo que permita acesso somente leitura a um bucket do Amazon S3. É possível, opcionalmente, armazenar informações de configuração ou um script de bootstrap nesse bucket, e os contêineres no Pod podem ler o arquivo do bucket e carregá-lo na aplicação. Se você quiser criar esse exemplo de política, copie o conteúdo a seguir para o seu dispositivo. Substitua *my-pod-secrets-bucket* pelo nome do seu bucket e execute o comando.

      ```
      cat >my-policy.json <<EOF
      {
          "Version": "2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Action": "s3:GetObject",
                  "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
              }
          ]
      }
      EOF
      ```

   b. Crie a política do IAM.

      ```
      aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
      ```

2. Crie um perfil do IAM e associe-o a uma conta de serviço do Kubernetes.

1. Se você já tem um conta de serviço do Kubernetes na qual você deseja assumir um perfil do IAM, você pode ignorar esta etapa.

   Criar uma conta de serviço do Kubernetes. Copie o conteúdo a seguir para o seu dispositivo. Substitua *my-service-account* pelo nome desejado e *default* (padrão) por um namespace diferente, se necessário. Se você alterar o *default* (padrão), o espaço de nome já deve existir.

   ```
   cat >my-service-account.yaml <<EOF
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: my-service-account
     namespace: default
   EOF
   kubectl apply -f my-service-account.yaml
   ```

   Execute o seguinte comando .

   ```
   kubectl apply -f my-service-account.yaml
   ```

2. Execute o seguinte comando para criar um arquivo de política de confiança para o perfil do IAM.

   ```
   cat >trust-relationship.json <<EOF
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
               "Effect": "Allow",
               "Principal": {
                   "Service": "pods.eks.amazonaws.com"
               },
               "Action": [
                   "sts:AssumeRole",
                   "sts:TagSession"
               ]
           }
       ]
   }
   ```

```
EOF
```

3. Crie a função. Substitua *my-role* por um nome para o perfil do IAM e *my-role-description* por uma descrição da função.

```
aws iam create-role --role-name my-role --assume-role-policy-document
 file://trust-relationship.json --description "my-role-description"
```

4. Associe uma política do IAM ao seu perfil. Substitua *my-role* pelo o nome de seu perfil do IAM e *my-policy* pelo nome de uma política existente que você tenha criado.

```
aws iam attach-role-policy --role-name my-role --policy-
arn=arn:aws:iam::111122223333:policy/my-policy
```

> **Note**
>
> Ao contrário dos perfis do IAM para contas de serviço, o EKS Pod Identity não usa uma anotação na conta de serviço.

5. Execute o comando a seguir para criar a associação. Substitua `my-cluster` pelo nome do cluster, substitua *my-service-account* pelo nome desejado e *default* por um namespace diferente, se necessário.

```
aws eks create-pod-identity-association --cluster-name my-cluster --role-
arn arn:aws:iam::111122223333:role/my-role --namespace default --service-
account my-service-account
```

Veja um exemplo de saída abaixo.

```
{
    "association": {
        "clusterName": "my-cluster",
        "namespace": "default",
        "serviceAccount": "my-service-account",
        "roleArn": "arn:aws:iam::111122223333:role/my-role",
        "associationArn": "arn:aws::111122223333:podidentityassociation/my-
cluster/a-abcdefghijklmnop1",
        "associationId": "a-abcdefghijklmnop1",
        "tags": {},
        "createdAt": 1700862734.922,
```

```
        "modifiedAt": 1700862734.922
    }
}
```

> **Note**
>
> É possível especificar um namespace e uma conta de serviço por nome que não existe no cluster. Você deve criar o namespace, a conta de serviço e a workload que usa a conta de serviço para que a associação ao EKS Pod Identity funcione.

3. Confirme se o perfil e a conta de serviço estão configuradas corretamente.

   a. Confirme se a política de confiança do perfil do IAM está configurada corretamente.

   ```
   aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
   ```

   Veja um exemplo de saída abaixo.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "Allow EKS Auth service to assume this role for Pod
    Identities",
               "Effect": "Allow",
               "Principal": {
                   "Service": "pods.eks.amazonaws.com"
               },
               "Action": [
                   "sts:AssumeRole",
                   "sts:TagSession"
               ]
           }
       ]
   }
   ```

   b. Confirme se a política que você anexou ao seu perfil em uma etapa anterior está vinculada ao perfil.

```
aws iam list-attached-role-policies --role-name my-role --query
 AttachedPolicies[].PolicyArn --output text
```

Veja um exemplo de saída abaixo.

```
arn:aws:iam::111122223333:policy/my-policy
```

c. Defina uma variável para armazenar o nome do recurso da Amazon (ARN) da política que deseja usar. Substitua *my-policy* pelo nome da política para a qual deseja confirmar permissões.

```
export policy_arn=arn:aws:iam::111122223333:policy/my-policy
```

d. Visualize a versão padrão da política.

```
aws iam get-policy --policy-arn $policy_arn
```

Veja um exemplo de saída abaixo.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws:iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

e. Revise o conteúdo da política para garantir que ela inclua todas as permissões de que seu Pod precisa. Se necessário, substitua*1* no comando a seguir pela versão retornada na saída anterior.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

Veja um exemplo de saída abaixo.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Se você criou a política de exemplo em uma etapa anterior, sua saída será a mesma. Se você criou uma política diferente, então o *exemplo* de conteúdo é diferente.

Próxima etapa

Configurar pods para acessar serviços da AWS com contas de serviço

Configurar pods para acessar serviços da AWS com contas de serviço

Se um Pod precisar acessar os Serviços da AWS, então você deverá configurá-lo para usar uma conta de serviço Kubernetes. A conta de serviço deve estar associada a um perfil do AWS Identity and Access Management (IAM) que tenha permissões para acessar Serviços da AWS.

Pré-requisitos

- Um cluster existente. Se você não tiver, poderá criar um, usando a CLI ou o console do Começar a usar o Amazon EKS.
- Uma associação entre a conta de serviço do Kubernetes e o EKS Pod Identity existente que associa a conta de serviço a um perfil do IAM. O perfil deve ter uma política do IAM associada que contenha as permissões que você deseja que seus Pods tenham para usar Serviços da AWS. Para obter mais informações sobre como criar a conta de serviço e como configurá-la, consulte Atribuir um perfil do IAM a uma conta de serviço do Kubernetes.
- A versão mais recente do AWS CLI instalada e configurada em seu dispositivo ou no AWS CloudShell. É possível verificar sua versão atual com `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o `aws configure` no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente.

Para atualizá-la, consulte Instalar a AWS CLI no diretório base no Guia do usuário do AWS CloudShell.

- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.
- Um arquivo `kubectl config` existente que contém a configuração do seu cluster. Para criar um arquivo `kubectl config`, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig.

Para configurar um Pod para usar uma conta de serviço

1. Use o comando a seguir para criar um manifesto de implantação com o qual é possível implantar um Pod para confirmar a configuração. Substitua *`example values`* pelos seus próprios valores.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Implante o manifesto no seu cluster.

```
kubectl apply -f my-deployment.yaml
```

3. Confirme se as variáveis de ambiente necessárias existem para seu Pod.

   a. Visualize os Pods que foram implantados na implantação da etapa anterior.

   ```
   kubectl get pods | grep my-app
   ```

   Veja um exemplo de saída abaixo.

   ```
   my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
   ```

   b. Verifique se o Pod tem um arquivo de token de conta de serviço montado.

   ```
   kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
    AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
   ```

   Veja um exemplo de saída abaixo.

   ```
   AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:  /var/run/secrets/
   pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
   ```

4. Confirme que seus Pods possam interagir com os Serviços da AWS usando as permissões que você atribuiu na política do IAM associada ao seu perfil.

> **Note**
>
> Quando um Pod usa credenciais da AWS de um perfil do IAM associado a uma conta de serviço, a AWS CLI ou outros SDKs nos contêineres desse Pod usam as credenciais fornecidas por esse perfil. Se você não restringir o acesso às credenciais fornecidas à perfil do IAM do nó do Amazon EKS, o Pod ainda terá acesso a essas credenciais. Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Se os seus Pods não conseguirem interagir com os serviços conforme o esperado, conclua as etapas a seguir para confirmar se tudo está configurado corretamente.

a. Confirme que seus Pods usam uma versão do AWS SDK que permita assumir um perfil do IAM por meio de uma associação ao EKS Pod Identity. Para ter mais informações, consulte Usar identidade de pods com o AWS SDK.

b. Confirme se a implantação está usando a conta de serviço.

```
kubectl describe deployment my-app | grep "Service Account"
```

Veja um exemplo de saída abaixo.

```
Service Account:  my-service-account
```

Conceder acesso aos pods a recursos da AWS baseados em tags

O EKS Identity Pod anexa tags às credenciais temporárias de cada pod com atributos como nome do cluster, namespace e nome da conta de serviço. Essas tags de sessão de perfil permitem que os administradores criem um único perfil que pode funcionar em várias contas de serviço, permitindo o acesso aos recursos da AWS com base em tags correspondentes. Ao adicionar suporte a tags de sessão de perfil, os clientes podem impor limites de segurança mais rígidos entre clusters e workloads dentro dos clusters enquanto reutilizam os mesmos perfis e políticas do IAM.

For exemplo, a política a seguir permitirá a ação `s3:GetObject` se o objeto estiver marcado com o nome do cluster do EKS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/eks-cluster-name": "${aws:PrincipalTag/eks-
cluster-name}"
                }
            }
        }
    ]
}
```

Lista de tags de sessão adicionadas pelo EKS Pod Identity

A lista a seguir contém todas as chaves das tags que são adicionadas à solicitação de `AssumeRole` feita pelo Amazon EKS. Para usar essas tags em políticas, use `${aws:PrincipalTag/` seguido pela chave, por exemplo, `${aws:PrincipalTag/kubernetes-namespace}`.

- `eks-cluster-arn`
- `eks-cluster-name`
- `kubernetes-namespace`
- `kubernetes-service-account`
- `kubernetes-pod-name`
- `kubernetes-pod-uid`

Cópia entre contas

Todas as tags de sessão adicionadas pelo EKS Pod Identity são transitivas; as chaves e os valores das tags são passados para quaisquer ações `AssumeRole` usadas por suas workloads para trocar de perfil para outra conta. Você pode usar essas tags em políticas de outras contas para limitar o acesso em cenários entre contas. Para obter mais informações, consulte Encadeamento de funções com tags de sessão no Guia do usuário do IAM.

Tags personalizadas

O EKS Pod Identity não pode adicionar outras tags personalizadas à ação `AssumeRole` que ele mesmo executa. No entanto, as tags que você aplica ao perfil do IAM estão sempre disponíveis no mesmo formato: `${aws:PrincipalTag/` seguidas pela chave, por exemplo, `${aws:PrincipalTag/MyCustomTag}`.

> **Note**
>
> As tags adicionadas à sessão por meio da solicitação `sts:AssumeRole` têm precedência em caso de conflito. Por exemplo, suponha que o Amazon EKS agregue uma chave `eks-cluster-name` e um valor `my-cluster` à sessão quando o EKS assume o perfil de cliente. Você também adicionou uma tag `eks-cluster-name` ao perfil do IAM com valor `my-own-cluster`. Nesse caso, o primeiro tem precedência e o valor da tag `eks-cluster-name` será `my-cluster`.

Usar identidade de pods com o AWS SDK

Usar as credenciais do EKS Pod Identity

Para usar as credenciais de uma associação ao EKS Pod Identity, seu código pode usar qualquer AWS SDK para criar um cliente para um serviço da AWS com um SDK e, por padrão, o SDK pesquisa em uma cadeia de locais as credenciais do AWS Identity and Access Management a serem usadas. As credenciais do EKS Pod Identity serão usadas se você não especificar um provedor de credenciais ao criar o cliente ou ao inicializar o SDK.

Isso funciona porque as identidades do EKS Pod foram adicionadas ao Provedor de credenciais do container, o qual é pesquisado em uma etapa na cadeia de credenciais padrão. Se suas workloads usam no momento credenciais mais antigas na cadeia de credenciais, essas credenciais continuarão sendo usadas mesmo se você configurar uma associação do EKS Pod Identity para a mesma workload.

Para obter mais informações sobre como as identidades do EKS Pod funcionam, consulte Entender como a EKS Pod Identity funciona.

Ao usar Saiba como a EKS Pod Identity concede aos pods acesso aos serviços da AWS, os contêineres nos Pods devem usar uma versão do AWS SDK compatível com o recurso de assumir um perfil do IAM do EKS Pod Identity Agent. Certifique-se de usar as seguintes versões, ou versões posteriores, do AWS SDK:

- Java (versão 2): 2.21.30
- Java: 1.12.746
- Go v1: v1.47.11
- Go v2: release-2023-11-14

- Python (Boto3) – 1.34.41
- Python (botocore): 1.34.41
- AWS CLI: 1.30.0

  AWS CLI: 2.15.0
- JavaScript v2: 2.1550.0
- JavaScript v3: 3.458.0
- Kotlin: v1.0.1
- Ruby: 3.188.0
- Rust: release-2024-03-13
- C++: 1.11.263
- .NET: 3.7.734.0
- PowerShell: 4.1.502
- PHP: 3.287.1

Para garantir que você esteja usando um SDK compatível, siga as instruções de instalação do SDK de sua preferência em Ferramentas para desenvolver na AWS quando estiver criando seus contêineres.

Para obter uma lista de complementos compatíveis com o EKS Pod Identity, consulte Versões de complementos compatíveis com o EKS Pod Identity.

Desabilitar o **`IPv6`** no EKS Pod Identity Agent

AWS Management Console

Desabilitar **`IPv6`** no AWS Management Console

1. Para desabilitar o `IPv6` no EKS Pod Identity Agent, adicione a configuração a seguir às Configurações opcionais do EKS Add-on.

   a. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

   b. No painel de navegação esquerdo, selecione Clusters e depois o nome do cluster para o qual você deseja configurar o complemento.

   c. Escolha a guia Add-ons (Complementos).

d. Selecione a caixa no canto superior direito da caixa do complemento do EKS Pod Identity Agent e escolha Editar.

e. Na página Configurar o EKS Pod Identity Agent:

i. Selecione a Version (Versão) que você deseja usar. Recomendamos manter a mesma versão da etapa anterior e atualizar a versão e a configuração em ações separadas.

ii. Expanda Definições de configuração opcionais.

iii. Insira a chave JSON `"agent":` e o valor de um objeto JSON aninhado com uma chave `"additionalArgs":` em Valores de configuração. O texto resultante deve ser um objeto JSON válido. Se esse par de chave e valor for o único dado na caixa de texto, coloque-o entre colchetes `{}`. O exemplo apresentado a seguir mostra que a política de rede está habilitada:

```
{
    "agent": {
        "additionalArgs": {
            "-b": "169.254.170.23"
        }
    }
}
```

Essa configuração define o endereço `IPv4` como o único usado pelo agente.

f. Para aplicar a nova configuração substituindo os pods do EKS Pod Identity Agent, escolha Salvar alterações.

O Amazon EKS aplica alterações nos complementos do EKS usando uma distribuição do Kubernetes `DaemonSet` para EKS Pod Identity Agent. É possível acompanhar o status da implantação no Histórico de atualizações do complemento no AWS Management Console e com `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`.

`kubectl rollout` oferece os seguintes comandos:

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
```

```
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Se a distribuição demorar muito, o Amazon EKS desfará a distribuição e uma mensagem com o tipo Atualização do complemento e o status Falha será adicionada ao Histórico de atualizações do complemento. Para investigar qualquer problema, comece com o histórico da distribuição e execute `kubectl logs` em um pod do EKS Pod Identity Agent para ver os logs do EKS Pod Identity Agent.

2. Se a nova entrada no Histórico de atualizações tiver o status de Êxito, a distribuição foi concluída e o complemento está usando a nova configuração em todos os pods do EKS Pod Identity Agent.

AWS CLI

Desabilitar **`IPv6`** no AWS CLI

- Para desabilitar o `IPv6` no EKS Pod Identity Agent, adicione a seguinte configuração aos valores de configuração do EKS Add-on.

  Execute o seguinte comando AWS CLI. Substitua `my-cluster` pelo nome do cluster e o ARN do perfil do IAM pelo perfil que você está usando.

  ```
  aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent \
      --resolve-conflicts PRESERVE --configuration-values '{"agent": {"additionalArgs": { "-b": "169.254.170.23"}}}'
  ```

  Essa configuração define o endereço `IPv4` como o único usado pelo agente.

  O Amazon EKS aplica alterações nos complementos do EKS usando uma distribuição do Kubernetes DaemonSet para EKS Pod Identity Agent. É possível acompanhar o status da implantação no Histórico de atualizações do complemento no AWS Management Console e com `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`.

`kubectl rollout` oferece os seguintes comandos:

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Se a distribuição demorar muito, o Amazon EKS desfará a distribuição e uma mensagem com o tipo Atualização do complemento e o status Falha será adicionada ao Histórico de atualizações do complemento. Para investigar qualquer problema, comece com o histórico da distribuição e execute `kubectl logs` em um pod do EKS Pod Identity Agent para ver os logs do EKS Pod Identity Agent.

Criar um perfil do IAM com a política de confiança exigida pela EKS Pod Identity

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`

O EKS Pod Identity usa `AssumeRole` para assumir o perfil do IAM antes de passar as credenciais temporárias para seus pods.

`sts:TagSession`

O EKS Pod Identity usa `TagSession` para incluir tags de sessão nas solicitações para AWS STS.

Você pode usar essas tags nas condition keys na política de confiança para restringir quais contas de serviço, namespaces e clusters podem usar esse perfil.

Para obter uma lista de chaves de condição do Amazon EKS, consulte Condições definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar a chave de condição, consulte Ações definidas pelo Amazon Elastic Kubernetes Service.

Perfis do IAM para contas de serviço

Aplicações nos contêineres de um Pod podem usar um AWS SDK ou AWS CLI para fazer solicitações de API aos Serviços da AWS usando permissões do AWS Identity and Access Management (IAM). As aplicações devem assinar suas solicitações de API da AWS com as credenciais da AWS. As funções do IAM para contas de serviço fornecem a capacidade de gerenciar credenciais para suas aplicações, semelhante à maneira como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2. Em vez de criar e distribuir suas credenciais da AWS aos contêineres ou de usar o perfil da instância do Amazon EC2, você pode associar um perfil do IAM a uma conta de serviço do Kubernetes e configurar os Pods para usar a conta de serviço. Não é possível usar perfis do IAM para contas de serviço com clusters locais do Amazon EKS no AWS Outposts.

Os perfis do IAM para contas de serviço oferecem as seguintes vantagens:

- Menor privilégio: é possível definir o escopo de permissões do IAM para uma conta de serviço, e somente os Pods que usarem essa conta de serviço terão acesso a essas permissões. Esse recurso também elimina a necessidade de soluções de terceiros, como o `kiam` ou o `kube2iam`.
- Isolamento de credenciais: um contêiner de Pod's só pode recuperar credenciais para o perfil do IAM que esteja associada à conta de serviço que o contêiner use. Um contêiner nunca tem acesso a credenciais usadas por outros contêineres em outros Pods. Ao usar perfis do IAM para contas

de serviço, os contêineres de Pod's também têm as permissões atribuídas ao , a menos que você bloqueie o acesso do ao PodServiço de metadados da instância do Amazon EC2 (IMDS). Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

- Auditabilidade:: o acesso e os logs de eventos estão disponíveis por meio do AWS CloudTrail para ajudar a garantir a auditoria retrospectiva.

Habilite perfis do IAM para contas de serviço concluindo os seguintes procedimentos:

1. Criar um provedor OIDC do IAM para o cluster: você só conclui este procedimento uma vez para cada cluster.

   > **Note**
   >
   > Se você habilitou o endpoint da VPC do EKS, o endpoint de serviço de OIDC do EKS não poderá ser acessado de dentro dessa VPC. Consequentemente, suas operações, como a criação de um provedor de OIDC `eksctl` dentro da VPC, não funcionarão e resultarão em um tempo limite ao tentar solicitar `https://oidc.eks.region.amazonaws.com`. Segue um exemplo de mensagem de erro:
   >
   > ```
   > ** server can't find oidc.eks.region.amazonaws.com: NXDOMAIN
   > ```
   >
   > Para concluir essa etapa, você pode executar o comando fora da VPC, por exemplo, em AWS CloudShell ou em um computador conectado à Internet. Como alternativa, você pode criar um resolvedor condicional de horizonte segmentado na VPC, como o Route 53 Resolver, para usar um resolvedor diferente para o URL do emissor do OIDC e não usar o DNS da VPC para ela. Para ver um exemplo de encaminhamento condicional em CoreDNS, consulte a Solicitação de recurso do Amazon EKS no GitHub.

2. Atribuir perfis do IAM às contas de serviço do Kubernetes: conclua este procedimento para cada conjunto exclusivo de permissões que você deseja que uma aplicação tenha.
3. Para configurar Pods para usar uma conta de serviço do Kubernetes: conclua este procedimento para cada Pod que precisa acessar Serviços da AWS.
4. Usar o IRSA com o AWS SDK: confirme se a workload usa um AWS SDK de uma versão compatível e se a workload usa a cadeia de credenciais padrão.

Informações básicas do IAM, Kubernetes e OpenID Connect (OIDC)

Em 2014, o AWS Identity and Access Management adicionou suporte para identidades federadas usando o OpenID Connect (OIDC). Esse recurso permite que você autentique chamadas de API da AWS com provedores de identidade compatíveis e receba um token de Web JSON OIDC válido (JWT). É possível passar esse token para a operação da API `AssumeRoleWithWebIdentity` do AWS STS e receber credenciais temporárias da função do IAM. É possível usar essas credenciais para interagir com qualquer Serviço da AWS, incluindo o Amazon S3 e o DynamoDB.

Cada token JWT é assinado por um par de chaves de assinatura. As chaves são servidas no provedor OIDC gerenciado pelo Amazon EKS e a chave privada é trocada a cada 7 dias. O Amazon EKS mantém as chaves públicas até a sua expiração. Se você conectar clientes OIDC externos, esteja ciente de que precisará atualizar as chaves de assinatura antes que a chave pública expire. Aprenda como the section called “Buscar chaves de assinatura”.

O Kubernetes há muito tempo usa contas de serviço como seu próprio sistema de identidade interno. Os Pods podem se autenticar com o servidor de API do Kubernetes usando um token montado automaticamente (que não era OIDC JWT) que apenas o servidor de API do Kubernetes pode validar. Esses tokens de conta de serviço legados não expiram, e a rotação da chave de assinatura é um processo difícil. Na versão `1.12` do Kubernetes, foi adicionado suporte para um novo recurso `ProjectedServiceAccountToken`. Esse recurso é um token da Web JSON OIDC que também contém a identidade da conta de serviço e oferece suporte a um público configurável.

O Amazon EKS hospeda um endpoint público de descoberta de OIDC em cada cluster que contém as chaves de assinatura para os tokens de Web JSON `ProjectedServiceAccountToken`, de modo que sistemas externos, como o IAM, possam validar e aceitar os tokens de OIDC emitidos pelo Kubernetes.

Criar um provedor OIDC do IAM para o cluster

Seu cluster tem um URL emissor do do OpenID Connect (OIDC) associada a ele. Para usar perfis do AWS Identity and Access Management (IAM) para contas de serviço, um provedor OIDC do IAM deve existir para a URL do emissor de OIDC do cluster.

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS.
- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para

verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.

- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.
- Um arquivo `kubectl config` existente que contém a configuração do seu cluster. Para criar um arquivo `kubectl config`, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig.

É possível criar um provedor OIDC do IAM para o cluster usando o `eksctl` ou o AWS Management Console.

eksctl

Pré-requisito

Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Para criar um provedor de identidade OIDC do IAM para o cluster com o **`eksctl`**

1. Determine o ID do emissor OIDC do seu cluster.

   Recupere o ID do emissor do OIDC do cluster e armazene-a em uma variável. Substitua *`my-cluster`* pelos seus próprios valores.

   ```
   cluster_name=my-cluster
   ```

```
oidc_id=$(aws eks describe-cluster --name $cluster_name --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

```
echo $oidc_id
```

2. Determine se um provedor do OIDC do IAM com seu ID do emissor do cluster já está em sua conta.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Se um resultado for retornado, significará que você já tem um provedor OIDC do IAM para o cluster e poderá pular a próxima etapa. Se nenhum resultado for retornado, você deverá criar um provedor OIDC do IAM para seu cluster.

3. Crie o provedor de identidade de OIDC do IAM para o cluster com o comando a seguir.

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
```

> **Note**
>
> Se você habilitou o endpoint da VPC do EKS, o endpoint de serviço de OIDC do EKS não poderá ser acessado de dentro dessa VPC. Consequentemente, suas operações, como a criação de um provedor de OIDC `eksctl` dentro da VPC, não funcionarão e resultarão em um tempo limite ao tentar solicitar `https://oidc.eks.region.amazonaws.com`. Segue um exemplo de mensagem de erro:
>
> ```
> ** server can't find oidc.eks.region.amazonaws.com: NXDOMAIN
> ```
>
> Para concluir essa etapa, você pode executar o comando fora da VPC, por exemplo, em AWS CloudShell ou em um computador conectado à Internet. Como alternativa, você pode criar um resolvedor condicional de horizonte segmentado na VPC, como o Route 53 Resolver, para usar um resolvedor diferente para o URL do emissor do OIDC e não usar o DNS da VPC para ela. Para ver um exemplo de encaminhamento condicional em CoreDNS, consulte a Solicitação de recurso do Amazon EKS no GitHub.

AWS Management Console

Para criar um provedor de identidade OIDC do IAM para o cluster com o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel esquerdo, selecione Clusters e, em seguida, selecione o nome do cluster na página Clusters.
3. Na seção Details (Detalhes) da guia Overview (Visão geral), observe o valor de OpenID Connect provider URL (URL do provedor OpenID Connect).
4. Abra o console do IAM em https://console.aws.amazon.com/iam/.
5. No painel de navegação à esquerda, escolha Identity Providers (Provedores de identidade) em Access management (Gerenciamento de acesso). Se um fornecedor listado corresponder à URL do cluster, então você já tem um provedor para o cluster. Se um provedor não estiver listado que corresponda à URL do seu cluster, você deverá criar um.
6. Para criar um provedor, selecione Add provider (Adicionar provedor).
7. Em Provider type (Tipo de provedor), selecione OpenID Connect.
8. Em Provider URL (URL do provedor), insira o URL do provedor OIDC do cluster e escolha Get thumbprint (Obter impressão digital).
9. Para Audience (Público), insira **`sts.amazonaws.com`** e escolha Add provider (Adicionar provedor).

Próxima etapa

Atribuir perfis do IAM às contas de serviço do Kubernetes

Atribuir perfis do IAM às contas de serviço do Kubernetes

Este tópico aborda como configurar uma conta de serviço do Kubernetes para assumir um perfil do AWS Identity and Access Management (IAM). Todos os Pods configurados para usar a conta de serviço podem então acessar quaisquer Serviço da AWS para os quais a função tenha permissões de acesso.

Pré-requisitos

- Um cluster existente. Se você não tiver, poderá criar um, seguindo um dos guias Começar a usar o Amazon EKS.

- Um provedor de OpenID Connect (OIDC) do IAM existente para o cluster. Para saber se você já tem um ou como criar um, consulte Criar um provedor OIDC do IAM para o cluster.
- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.
- Um arquivo `kubectl config` existente que contém a configuração do seu cluster. Para criar um arquivo `kubectl config`, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig.

Para associar um perfil do IAM a uma conta de serviço do Kubernetes.

1. Se você quiser associar uma política do IAM existente ao seu perfil do IAM, vá para a próxima etapa.

   Crie uma política do IAM. É possível criar a sua própria política ou copiar uma política gerenciada da AWS que já conceda algumas das permissões de que você precisa e a personalizar de acordo com seus requisitos específicos. Para obter mais informações, consulte Criar políticas do IAM no Guia do usuário do IAM.

   a. Crie um arquivo que inclua as permissões para os Serviços da AWS que você deseja que seus Pods acessem. Para obter uma lista de todas as ações para todos os Serviços da AWS, consulte a Referência de autorização do serviço.

      É possível executar o comando a seguir para criar um arquivo de política de exemplo que permita acesso somente leitura a um bucket do Amazon S3. É possível, opcionalmente, armazenar informações de configuração ou um script de bootstrap nesse bucket, e os

contêineres no Pod podem ler o arquivo do bucket e carregá-lo na aplicação. Se você quiser criar esse exemplo de política, copie o conteúdo a seguir para o seu dispositivo. Substitua *my-pod-secrets-bucket* pelo nome do seu bucket e execute o comando.

```
cat >my-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
EOF
```

b. Crie a política do IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

2. Crie um perfil do IAM e associe-o a uma conta de serviço do Kubernetes. Você pode usar o `eksctl` ou a AWS CLI.

eksctl

Pré-requisito

Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Substitua *my-service-account* pelo nome da conta de serviço do Kubernetes Kubernetes que você deseja que o `eksctl` crie e associe a um perfil do IAM. Substitua *default* (padrão) pelo namespace em que você deseja que o `eksctl` crie a conta de serviço. Substitua *my-cluster* pelo nome do cluster. Substitua *my-role* pelo nome da função à qual você deseja associar a conta de serviço. Se ela ainda não existir, o `eksctl` a criará para você. Substitua *111122223333* pelo ID da sua conta e *my-policy* pelo nome de uma política existente.

```
eksctl create iamserviceaccount --name my-service-account --namespace default --
cluster my-cluster --role-name my-role \
    --attach-policy-arn arn:aws:iam::111122223333:policy/my-policy --approve
```

> **Important**
>
> Se a função ou a conta de serviço já existir, o comando anterior poderá falhar. O `eksctl` tem opções diferentes que você pode oferecer nessas situações. Para obter mais informações, execute **`eksctl create iamserviceaccount --help`**.

AWS CLI

1. Se você já tem um conta de serviço do Kubernetes na qual você deseja assumir um perfil do IAM, você pode ignorar esta etapa.

   Criar uma conta de serviço do Kubernetes. Copie o conteúdo a seguir para o seu dispositivo. Substitua *`my-service-account`* pelo nome desejado e *`default`* (padrão) por um namespace diferente, se necessário. Se você alterar o *`default`* (padrão), o espaço de nome já deve existir.

   ```
   cat >my-service-account.yaml <<EOF
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: my-service-account
     namespace: default
   EOF
   kubectl apply -f my-service-account.yaml
   ```

2. Defina o ID da Conta da AWS como uma variável de ambiente com o comando a seguir.

   ```
   account_id=$(aws sts get-caller-identity --query "Account" --output text)
   ```

3. Defina o provedor de identidade OIDC do seu cluster como uma variável de ambiente com o comando a seguir. Substitua o *`my-cluster`* pelo nome do cluster.

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region
 $AWS_REGION --query "cluster.identity.oidc.issuer" --output text | sed -e "s/
^https:\/\///")
```

4. Defina variáveis para o namespace e o nome da conta de serviço. Substitua *my-service-account* e pelo serviço do Kubernetes e a conta de serviço que você deseja associar ao perfil. Substitua *default* (padrão) pelo namespace da conta de serviço.

```
export namespace=default
export service_account=my-service-account
```

5. Execute o seguinte comando para criar um arquivo de política de confiança para o perfil do IAM. Se você quiser permitir que todas as contas de serviço em um namespace usem o perfil, copie o conteúdo a seguir para o seu dispositivo. Substitua *StringEquals* por **StringLike** e substitua *$service_account* por *****. É possível adicionar várias entradas nas condições `StringEquals` ou `StringLike` abaixo para permitir que várias contas de serviço ou namespaces assumam o perfil. Para permitir perfis uma Conta da AWS diferente da conta em que seu cluster está para assumir o perfil, consulte Autenticar em outra conta com IRSA para obter mais informações.

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:
$namespace:$service_account"
        }
      }
    }
  ]
}
```

```
EOF
```

6. Crie a função. Substitua *my-role* por um nome para o perfil do IAM e *my-role-description* por uma descrição da função.

```
aws iam create-role --role-name my-role --assume-role-policy-document
 file://trust-relationship.json --description "my-role-description"
```

7. Associe uma política do IAM ao seu perfil. Substitua *my-role* pelo o nome de seu perfil do IAM e *my-policy* pelo nome de uma política existente que você tenha criado.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:iam::
$account_id:policy/my-policy
```

8. Anote sua conta de serviço com o nome do recurso da Amazon (ARN) do perfil do IAM que você deseja que a conta de serviço assuma. Substitua *my-role* pelo nome do seu perfil do IAM existente. Suponha que você tenha permitido um perfil de outra Conta da AWS diferente da conta em que o cluster está para assumir o perfil em uma etapa anterior. Em seguida, especifique a Conta da AWS e o perfil da outra conta. Para ter mais informações, consulte Autenticar em outra conta com IRSA.

```
kubectl annotate serviceaccount -n $namespace $service_account
 eks.amazonaws.com/role-arn=arn:aws:iam::$account_id:role/my-role
```

3. Confirme se o perfil e a conta de serviço estão configuradas corretamente.

   a. Confirme se a política de confiança do perfil do IAM está configurada corretamente.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

   Veja um exemplo de saída abaixo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
```

```
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-
service-account",
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
}
```

b. Confirme se a política que você anexou ao seu perfil em uma etapa anterior está vinculada ao perfil.

```
aws iam list-attached-role-policies --role-name my-role --query
 AttachedPolicies[].PolicyArn --output text
```

Veja um exemplo de saída abaixo.

```
arn:aws:iam::111122223333:policy/my-policy
```

c. Defina uma variável para armazenar o nome do recurso da Amazon (ARN) da política que deseja usar. Substitua *my-policy* pelo nome da política para a qual deseja confirmar permissões.

```
export policy_arn=arn:aws:iam::111122223333:policy/my-policy
```

d. Visualize a versão padrão da política.

```
aws iam get-policy --policy-arn $policy_arn
```

Veja um exemplo de saída abaixo.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws:iam::111122223333:policy/my-policy",
```

```
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

e. Revise o conteúdo da política para garantir que ela inclua todas as permissões de que seu Pod precisa. Se necessário, substitua*1* no comando a seguir pela versão retornada na saída anterior.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

Veja um exemplo de saída abaixo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Se você criou a política de exemplo em uma etapa anterior, sua saída será a mesma. Se você criou uma política diferente, então o *exemplo* de conteúdo é diferente.

f. Confirme que a conta do serviço do Kubernetes esteja anotada com o perfil.

```
kubectl describe serviceaccount my-service-account -n default
```

Veja um exemplo de saída abaixo.

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com/role-arn:
 arn:aws:iam::111122223333:role/my-role
Image pull secrets:  <none>
Mountable secrets:   my-service-account-token-qqjfl
Tokens:              my-service-account-token-qqjfl
```

```
[...]
```

4. (Opcional) Configurar o endpoint do AWS Security Token Service para uma conta de serviço. A AWS recomenda usar um endpoint AWS STS regional em vez do endpoint global. Isso reduz a latência, fornece redundância integrada e aumenta a validade do token da sessão.

Próxima etapa

Para configurar Pods para usar uma conta de serviço do Kubernetes

Para configurar Pods para usar uma conta de serviço do Kubernetes

Se um Pod precisar acessar os Serviços da AWS, então você deverá configurá-lo para usar uma conta de serviço Kubernetes. A conta de serviço deve estar associada a um perfil do AWS Identity and Access Management (IAM) que tenha permissões para acessar Serviços da AWS.

Pré-requisitos

- Um cluster existente. Se você não tiver, poderá criar um, usando a CLI ou o console do Começar a usar o Amazon EKS.
- Um provedor de OpenID Connect (OIDC) do IAM existente para o cluster. Para saber se você já tem um ou como criar um, consulte Criar um provedor OIDC do IAM para o cluster.
- Um serviço Kubernetes existente que é associado a um perfil do IAM. A conta de serviço deve ser anotada com o nome do recurso da Amazon (ARN) do perfil do IAM. O perfil deve ter uma política do IAM associada que contenha as permissões que você deseja que seus Pods tenham para usar Serviços da AWS. Para obter mais informações sobre como criar a conta de serviço e como configurá-la, consulte Atribuir perfis do IAM às contas de serviço do Kubernetes.
- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à

versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.

- Um arquivo `kubectl config` existente que contém a configuração do seu cluster. Para criar um arquivo `kubectl config`, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig.

Para configurar um Pod para usar uma conta de serviço

1. Use o comando a seguir para criar um manifesto de implantação com o qual é possível implantar um Pod para confirmar a configuração. Substitua *`example values`* pelos seus próprios valores.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Implante o manifesto no seu cluster.

```
kubectl apply -f my-deployment.yaml
```

3. Confirme se as variáveis de ambiente necessárias existem para seu Pod.

   a. Visualize os Pods que foram implantados na implantação da etapa anterior.

```
kubectl get pods | grep my-app
```

Veja um exemplo de saída abaixo.

```
my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
```

b. Visualize o ARN do perfil do IAM que o Pod está usando.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
```

Veja um exemplo de saída abaixo.

```
AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/my-role
```

O ARN do perfil deve corresponder ao ARN do perfil com o qual você anotou a conta de serviço existente. Para saber mais sobre como fazer anotações na conta de serviço, consulte Atribuir perfis do IAM às contas de serviço do Kubernetes.

c. Confirme que o Pod tenha um arquivo de token de identidade da Web montado.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
 AWS_WEB_IDENTITY_TOKEN_FILE:
```

Veja um exemplo de saída abaixo.

```
AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

O `kubelet` solicita e armazena o token em nome do Pod. Por padrão, o `kubelet` atualiza o token se ele for mais antigo do que 80% do seu tempo de vida total, ou se o token tiver mais de 24 horas. É possível modificar a duração da expiração de qualquer conta, exceto a conta de serviço padrão, usando as configurações na especificação do Pod. Para obter mais informações, consulte Service Account Token Volume Projection (Projeção do volume de tokens da conta de serviço) na documentação do Kubernetes.

O Webhook de identidade do pod do Amazon EKS no cluster observa os Pods que usam uma conta de serviço com a seguinte anotação:

```
eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/my-role
```

O webhook aplica as variáveis de ambiente anteriores a esses Pods. Seu cluster não precisa usar o webhook para configurar as variáveis de ambiente e montagens de arquivos do token. É possível configurar manualmente o Pods para ter essas variáveis de ambiente. As versões compatíveis do SDK da AWS procuram essas variáveis de ambiente primeiro no fornecedor da cadeia de credenciais. As credenciais da função são usadas para os Pods que atendam a esses critérios.

4. Confirme que seus Pods possam interagir com os Serviços da AWS usando as permissões que você atribuiu na política do IAM associada ao seu perfil.

> **Note**
>
> Quando um Pod usa credenciais da AWS de um perfil do IAM associado a uma conta de serviço, a AWS CLI ou outros SDKs nos contêineres desse Pod usam as credenciais fornecidas por esse perfil. Se você não restringir o acesso às credenciais fornecidas à perfil do IAM do nó do Amazon EKS, o Pod ainda terá acesso a essas credenciais. Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Se os seus Pods não conseguirem interagir com os serviços conforme o esperado, conclua as etapas a seguir para confirmar se tudo está configurado corretamente.

a. Confirme que seus Pods usam uma versão do SDK AWS da que ofereça suporte a assumir um perfil do IAM por meio de um arquivo de token de identidade da Web OpenID Connect. Para ter mais informações, consulte Usar o IRSA com o AWS SDK.

b. Confirme se a implantação está usando a conta de serviço.

```
kubectl describe deployment my-app | grep "Service Account"
```

Veja um exemplo de saída abaixo.

```
Service Account:  my-service-account
```

c. Se os seus Pods ainda não conseguirem acessar os serviços, revise as etapas que estão descritas em Atribuir perfis do IAM às contas de serviço do Kubernetes para confirmar se seu perfil e conta de serviço estão configurados corretamente.

Configurar o endpoint do AWS Security Token Service para uma conta de serviço

Se você estiver usando uma conta de serviço do Kubernetes com o Perfis do IAM para contas de serviço, poderá configurar o tipo de endpoint do AWS Security Token Service usado pela conta de serviço se a versão do cluster e da plataforma forem as versões listadas na tabela a seguir ou versões posteriores. Se a versão do Kubernetes ou da plataforma forem anteriores às listadas na tabela, as contas de serviço só poderão usar o endpoint global.

Versão do Kubernetes	Versão da plataforma	Tipo de endpoint padrão
1.30	eks.2	Regional
1.29	eks.1	Regional
1.28	eks.1	Regional
1.27	eks.1	Regional
1.26	eks.1	Regional
1.25	eks.1	Regional
1.24	eks.2	Regional
1.23	eks.1	Regional

A AWS recomenda usar os endpoints AWS STS regionais em vez do endpoint global. Isso reduz a latência, fornece redundância integrada e aumenta a validade do token da sessão. O AWS Security Token Service deve estar ativo na Região da AWS em que o Pod estiver em execução. Além disso, sua aplicação deve ter redundância incorporada para uma Região da AWS diferente em caso de falha de serviço na Região da AWS. Para obter mais informações, consulte Gerenciar o AWS STS em uma Região da AWS no Guia do usuário do IAM.

Pré-requisitos

- Um cluster existente. Se você não tiver, poderá criar um, usando a CLI ou o console do Começar a usar o Amazon EKS.
- Um provedor de OIDC do IAM existente para o cluster. Para ter mais informações, consulte Criar um provedor OIDC do IAM para o cluster.
- Uma conta de serviço do Kubernetes existente configurada para uso com o recurso IAM para contas de serviço no Amazon EKS.

Para configurar o tipo de endpoint usado por uma conta de serviço do Kubernetes

Todos os exemplos a seguir usam a conta de serviço do Kubernetes do `aws-node` usada pelo plug-in Amazon VPC CNI. É possível substituir os *`example values`* por suas próprias contas de serviço, Pods, namespaces e outros recursos.

1. Selecione um Pod que use uma conta de serviço cujo endpoint você queira alterar. Determine em quais Região da AWS o Pod será executado. Substitua *`aws-node-6mfgv`* pelo nome do Pod e *`kube-system`* pelo namespace do Pod.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
   ```

   Veja um exemplo de saída abaixo.

   ```
   ip-192-168-79-166.us-west-2/192.168.79.166
   ```

   Na saída anterior, o Pod está sendo executado em um nó na Região da AWS us-west-2.
2. Determine o tipo de endpoint que a conta de serviço do Pod's está usando.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
   ```

   Veja um exemplo de saída abaixo.

   ```
   AWS_STS_REGIONAL_ENDPOINTS: regional
   ```

   Se o endpoint atual for global, `global` será retornado na saída. Se nenhuma saída for retornada, significa que tipo de endpoint padrão está em uso e não foi substituído.

3. Se a versão do cluster ou plataforma for a mesma ou posterior à da tabela, você poderá alterar o tipo de endpoint utilizado pela conta de serviço do tipo padrão para um tipo diferente, usando um dos comandos a seguir. Substitua *aws-node* pelo nome da conta de serviço e *kube-system* pelo namespace da conta de serviço.

   - Se o tipo de endpoint padrão ou atual for global e você quiser modificá-lo para regional:

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
     ```

     Se estiver usando Perfis do IAM para contas de serviço para gerar URLs do S3 pré-assinados na aplicação em execução em contêineres de Pods, o formato do URL para endpoints regionais será semelhante ao seguinte exemplo:

     ```
     https://bucket.s3.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
     ```

   - Se o tipo de endpoint padrão ou atual for global, e você quiser modificá-lo para regional:

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
     ```

     Se a aplicação estiver fazendo solicitações explicitamente para endpoints globais do AWS STS e você não substituir o comportamento padrão do uso de endpoints regionais em clusters do Amazon EKS, as solicitações falharão com um erro. Para ter mais informações, consulte Os contêineres do pod recebem o seguinte erro: `An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`.

     Se você estiver usando Perfis do IAM para contas de serviço para gerar URLs do S3 pré-assinadas na sua aplicação em execução em contêineres de Pods, o formato do URL para endpoints globais será semelhante ao do seguinte exemplo:

     ```
     https://bucket.s3.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
     ```

   Se você tiver uma automação que espera o URL pré-assinado em um determinado formato, ou se a aplicação ou as dependências downstream que utilizam URLs pré-assinadas tiverem

expectativas para a Região da AWS direcionada, faça as alterações necessárias para utilizar o endpoint do AWS STS apropriado.

4. Exclua e recrie os Pods existentes associados à conta de serviço para aplicar as variáveis de ambiente de credenciais. O webhook de mutação não as aplica aos Pods que já estão em execução. É possível substituir *Pods*, *kube-system* e *-l k8s-app=aws-node* pelas informações dos Pods para os quais você definiu a anotação.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

5. Confirme se todos os Pods foram reiniciados.

```
kubectl get Pods -n kube-system -l k8s-app=aws-node
```

6. Visualize as variáveis de ambiente de um dos Pods. Verifique se o valor `AWS_STS_REGIONAL_ENDPOINTS` é o que você o definiu em uma etapa anterior.

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

Veja um exemplo de saída abaixo.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Autenticar em outra conta com IRSA

É possível configurar permissões entre contas do IAM criando um provedor de identidade do cluster de outra conta ou usando operações `AssumeRole` encadeadas. Nos exemplos a seguir, a Conta A tem um cluster do Amazon EKS que oferece suporte a perfis do IAM para contas de serviço. Os Pods que estão em execução nesse cluster devem assumir permissões do IAM da Conta B.

Example Criar um provedor de identidade de cluster de outra conta

Example

Neste exemplo, a Conta A fornece à Conta B o URL do emissor de OpenID Connect (OIDC) de seu cluster. A Conta B segue as instruções em Criar um provedor OIDC do IAM para o cluster e Atribuir perfis do IAM às contas de serviço do Kubernetes usando o URL emissor OIDC do cluster da Conta A. Em seguida, um administrador de cluster anota a conta de serviço no cluster da Conta A para usar o perfil da Conta B (*444455556666*).

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::444455556666:role/account-b-role
```

Example Usar operações **`AssumeRole`** encadeadas

Example

Neste exemplo, a Conta B cria uma política do IAM com as permissões a serem concedidas aos Pods no cluster da Conta A. A Conta B (*444455556666*) anexa essa política a um perfil do IAM com uma relação de confiança que concede permissões `AssumeRole` à Conta A (*111122223333*).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

A conta A cria um perfil com uma política de confiança que obtém credenciais do provedor de identidade criado com o endereço do emissor de OIDC do cluster.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity"
    }
```

```
  ]
}
```

A Conta A anexa uma política a essa função com as seguintes permissões para assumir a função criada pela Conta B.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::444455556666:role/account-b-role"
        }
    ]
}
```

A aplicação cria o código para que os Pods assumam que a função da Conta B usa dois perfis: `account_b_role` e `account_a_role`. O perfil `account_b_role` usa o perfil `account_a_role` como origem. Para a AWS CLI, o arquivo `~/.aws/config` é semelhante ao exemplo a seguir.

```
[profile account_b_role]
source_profile = account_a_role
role_arn=arn:aws:iam::444455556666:role/account-b-role

[profile account_a_role]
web_identity_token_file = /var/run/secrets/eks.amazonaws.com/serviceaccount/token
role_arn=arn:aws:iam::111122223333:role/account-a-role
```

Para especificar perfis encadeados para outras SDKs da AWS, consulte a documentação do SDK em uso. Para obter mais informações, consulte Ferramentas para criar na AWS.

Usar o IRSA com o AWS SDK

Usar as credenciais

Para usar as credenciais de perfis do IAM para contas de serviço, seu código pode usar qualquer AWS SDK para criar um cliente para um serviço da AWS com um SDK e, por padrão, o SDK pesquisa em uma cadeia de locais as credenciais do AWS Identity and Access Management a serem usadas. As credenciais do perfis do IAM para contas de serviço serão usadas se você não especificar um provedor de credenciais ao criar o cliente ou ao inicializar o SDK.

Isso funciona porque os perfis do IAM para contas de serviço foram adicionados como uma etapa na cadeia de credenciais padrão. Se suas workloads usam no momento credenciais mais antigas na cadeia de credenciais, essas credenciais continuarão sendo usadas mesmo se você configurar perfis do IAM para contas de serviço para a mesma workload.

O SDK troca automaticamente o token OIDC da conta de serviço por credenciais temporárias do AWS Security Token Service usando a ação `AssumeRoleWithWebIdentity`. O Amazon EKS e essa ação do SDK continuam alternando as credenciais temporárias, renovando-as antes que elas expirem.

Ao usar Perfis do IAM para contas de serviço, os contêineres nos Pods devem usar uma versão do AWS SDK compatível com o recurso de assumir um perfil do IAM por meio de um arquivo de token de identidade de Web OpenID Connect. Certifique-se de usar as seguintes versões, ou versões posteriores, do AWS SDK:

- Java (versão 2): 2.10.11
- Java: 1.11.704
- Go: 1.23.13
- Python (Boto3): 1.9.220
- Python (botocore): 1.12.200
- AWS CLI: 1.16.232
- Node: 2.525.0 e 3.27.0
- Ruby: 3.58.0
- C++: 1.7.174
- .NET: 3.3.659.1 - Você também deve incluir o `AWSSDK.SecurityToken`.
- PHP: 3.110.7

Muitos complementos populares do Kubernetes, como o Cluster Autoscaler, o O que é o AWS Load Balancer Controller? e o Amazon VPC CNI plugin for Kubernetes, são compatíveis com os perfis do IAM para contas de serviço.

Para garantir que você esteja usando um SDK compatível, siga as instruções de instalação do SDK de sua preferência em Ferramentas para desenvolver na AWS quando estiver criando seus contêineres.

Obter as chaves de assinatura para validar os tokens OIDC

O Kubernetes emite um `ProjectedServiceAccountToken` para cada Service Account do Kubernetes. Este token é um token OIDC, que também é um tipo de JSON web token (JWT). O Amazon EKS hospeda um endpoint público OIDC em cada cluster que contém as chaves de assinatura para o token de modo que sistemas externos possam validá-lo.

Para validar um `ProjectedServiceAccountToken`, você precisa buscar as chaves públicas de assinatura OIDC, também chamadas de JSON Web Key Set (JWKS). Use essas chaves em sua aplicação para validar o token. Por exemplo, você pode usar a biblioteca PyJWT Python para validar tokens usando essas chaves. Para obter mais informações sobre o `ProjectedServiceAccountToken`, consulte o the section called “Informações básicas do IAM, Kubernetes e OpenID Connect (OIDC)”.

Pré-requisitos

- Um provedor OpenID Connect (OIDC) do AWS Identity and Access Management (IAM) existente para o cluster. Para determinar se você tem ou para criar uma, consulte Criar um provedor OIDC do IAM para o cluster.
- AWS CLI – Uma ferramenta de linha de comando para trabalhar com os serviços AWS, incluindo o Amazon EKS. Para obter mais informações, consulte Installing, updating, and uninstalling the AWS CLI (Instalar, atualizar e desinstalar a) no Manual do usuário da AWS Command Line Interface. Depois de instalar a AWS CLI, recomendamos que você também a configure. Para obter mais informações, consulte Quick configuration with `aws configure` (Configuração rápida com) no Manual do usuário do AWS Command Line Interface.

Buscar chaves públicas de assinatura OIDC (AWS CLI)

1. Recupere o URL do OIDC do seu cluster do Amazon EKS usando a AWS CLI.

```
$ aws eks describe-cluster --name my-cluster --query 'cluster.identity.oidc.issuer'
"https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE"
```

2. Recupere a chave de assinatura pública usando o curl ou uma ferramenta similar. O resultado é um JSON Web Key Set (JWKS).

> **Important**
>
> O Amazon EKS controla a utilização das chamadas para o endpoint do OIDC. Você deve armazenar em cache a chave pública de assinatura. Respeite o cabeçalho `cache-control` incluído na resposta.

> **Important**
>
> O Amazon EKS alterna a chave de assinatura OIDC a cada sete dias.

```
$ curl https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE/keys
{"keys":
[{"kty":"RSA","kid":"2284XXXX4a40","use":"sig","alg":"RS256","n":"wklbXXXXMVfQ","e":"AQAB"}
```

Gerenciar recursos computacionais usando nós

Um nó do Kubernetes é uma máquina que executa aplicações conteinerizadas. Todo nó tem os seguintes componentes:

- Runtime de contêiner: software responsável pela execução dos contêineres.
- **`kubelet`**: garante a integridade e o funcionamento dos contêineres dentro do Pod associado.
- **`kube-proxy`**: mantém regras de rede que permitem a comunicação com os Pods.

Para obter mais informações, consulte Nodes (Nós) na documentação do Kubernetes.

O cluster do Amazon EKS pode agendar Pods em qualquer combinação de nós autogerenciados, grupos de nós gerenciados do Amazon EKS e AWS Fargate. Para saber mais sobre nós implantados em seu cluster, consulte Visualizar os recursos do Kubernetes.

> **Important**
>
> O AWS Fargate com o Amazon EKS não está disponível para a AWS GovCloud (Leste dos EUA) e AWS GovCloud (Oeste dos EUA).

> **Note**
>
> Os nós devem estar na mesma VPC que as sub-redes selecionadas na ocasião da criação do cluster. Porém, esses nós não precisam estar nas mesmas sub-redes.

A tabela a seguir fornece vários critérios para avaliar ao decidir quais opções atendem melhor às suas necessidades. Esta tabela não inclui nós conectados que foram criados fora do Amazon EKS, que só podem ser visualizadas.

> **Note**
>
> O Bottlerocket tem algumas diferenças específicas em relação às informações gerais desta tabela. Para obter mais informações, consulte a documentação do Bottlerocket no GitHub.

Critérios	Grupos de nós gerenciados do EKS	Nós autogerenciados	AWS Fargate
Pode ser implantado no AWS Outposts	Não	Sim	Não
Pode ser implantado em uma Zona local da AWS	Não	Sim. Para obter mais informações, consulte Amazon EKS e zonas locais da AWS.	Não
Pode executar contêineres que exijam o Windows	Sim	Sim: mas, o cluster ainda requer, pelo menos, um nó Linux (dois são recomendáveis para garantir disponibilidade).	Não
Pode executar contêineres que exijam o Linux	Sim	Sim	Sim
Pode executar workloads que exijam o chip do Inferentia	Sim. Somente nós do Amazon Linux	Sim. Somente Amazon Linux	Não
Pode executar workloads que exijam uma GPU	Sim. Somente nós do Amazon Linux	Sim. Somente Amazon Linux	Não
Pode executar workloads que exijam processadores ARM	Sim	Sim	Não
Pode executar o AWS Bottlerocket	Sim	Sim	Não

Critérios	Grupos de nós gerenciados do EKS	Nós autogerenciados	AWS Fargate
Pods compartilham um ambiente de runtime do kernel com outros Pods	Sim. Todos os Pods em cada um dos nós	Sim. Todos os Pods em cada um dos nós	Não. Cada Pod tem um kernel dedicado
Os pods compartilham recursos de CPU, memória, armazenamento e rede com outros Pods.	Sim. Pode resultar em recursos não utilizados em cada nó	Sim. Pode resultar em recursos não utilizados em cada nó	Não. Cada Pod tem recursos dedicados e pode ser dimensionado independentemente para maximizar a utilização dos recursos.
Os pods podem usar mais hardware e memória do que o solicitado nas especificações do Pod	Sim. Se o Pod exigir mais recursos do que o solicitado e os recursos estiverem disponíveis no nó, o Pod poderá usar recursos adicionais.	Sim. Se o Pod exigir mais recursos do que o solicitado e os recursos estiverem disponíveis no nó, o Pod poderá usar recursos adicionais.	Não. Porém, o Pod pode ser reimplantado usando uma configuração maior de vCPU e memória.

Critérios	Grupos de nós gerenciados do EKS	Nós autogerenciados	AWS Fargate
Deve implantar e gerenciar instâncias do Amazon EC2	Sim. Automatizado pelo Amazon EKS se você implantou uma AMI otimizada do Amazon EKS Se você implantou uma AMI personalizada, você deve atualizar a instância manualmente.	Sim. Configuração manual ou uso de modelos AWS CloudFormation fornecidos pelo Amazon EKS para implantar nós Linux (x86), Linux (Arm) ou Windows.	Não
Deve proteger, manter e corrigir o sistema operacional das instâncias do Amazon EC2	Sim	Sim	Não
Pode fornecer argumentos de inicialização na implantação de um nó, como argumentos de `kubelet` extras.	Sim. Ao usar `eksctl` ou um modelo de execução com uma AMI personalizada.	Sim. Para obter mais informações, consulte as informações sobre o uso do script de bootstrap, no GitHub.	Não

Critérios	Grupos de nós gerenciados do EKS	Nós autogerenciados	AWS Fargate
Pode atribuir endereços IP a Pods de um bloco CIDR diferente do endereço IP atribuído ao nó.	Sim. Usando um modelo de execução com uma AMI personalizada Para obter mais informações, consulte Personalizar nós gerenciados com modelos de execução.	Sim. Para obter mais informações, consulte Rede personalizada para pods.	Não
É possível executar o SSH no nó	Sim	Sim	Não: não há nenhum sistema operacional de host de nó para SSH.
Pode implantar sua própria AMI personalizada em nós	Sim. Usar um modelo de execução	Sim	Não
Pode implantar seu próprio CNI personalizado em nós	Sim. Usando um modelo de execução com uma AMI personalizada	Sim	Não

Critérios	Grupos de nós gerenciados do EKS	Nós autogerenciados	AWS Fargate
Você deve atualizar a AMI do nó por conta própria	Yes (Sim): se você implantou uma AMI otimizada do Amazon EKS, receberá uma notificação no console do Amazon EKS quando as atualizações estiverem disponíveis. É possível executar a atualização com um clique no console. Se você implantou uma AMI personalizada, não será notificado no console do Amazon EKS quando as atualizações estiverem disponíveis. Você deve executar a atualização por conta própria.	Yes (Sim): usar ferramentas que não sejam do console do Amazon EKS. Isso ocorre porque nós autogerenciados não podem ser gerenciados com o console do Amazon EKS.	Não

Critérios	Grupos de nós gerenciados do EKS	Nós autogerenciados	AWS Fargate
É necessário atualizar a versão Kubernetes do nó por conta própria	Yes (Sim): se você implantou uma AMI otimizada do Amazon EKS, receberá uma notificação no console do Amazon EKS quando as atualizações estiverem disponíveis. É possível executar a atualização com um clique no console. Se você implantou uma AMI personalizada, não será notificado no console do Amazon EKS quando as atualizações estiverem disponíveis. Você deve executar a atualização por conta própria.	Yes (Sim): usar ferramentas que não sejam do console do Amazon EKS. Isso ocorre porque nós autogerenciados não podem ser gerenciados com o console do Amazon EKS.	Não. Você não gerencia nós.

Critérios	Grupos de nós gerenciados do EKS	Nós autogerenciados	AWS Fargate
Pode usar o armazenamento do Amazon EBS com os Pods	Sim	Sim	Não
Pode usar o armazenamento do Amazon EFS com os Pods	Sim	Sim	Sim
Pode usar o armazenamento do Amazon FSx para Lustre com os Pods	Sim	Sim	Não
Pode usar o Network Load Balancer para serviços	Sim	Sim	Sim, quando usar o Criar um balanceador de carga da rede
Pods podem ser executados em uma sub-rede pública	Sim	Sim	Não
Pode atribuir grupos de segurança da VPC a Pods individuais	Sim: somente nós do Linux	Sim: somente nós do Linux	Sim
Pode executar o Kubernetes DaemonSets	Sim	Sim	Não
Compatível com `HostPort` e `HostNetwork` no manifesto do Pod	Sim	Sim	Não
Disponibilidade do Região da AWS	Todas as regiões compatíveis com Amazon EKS	Todas as regiões compatíveis com Amazon EKS	Algumas regiões compatíveis com o Amazon EKS
Pode executar contêineres em hosts dedicados do Amazon EC2	Sim	Sim	Não

Critérios	Grupos de nós gerenciados do EKS	Nós autogerenciados	AWS Fargate
Definição de preço	Custo de instância do Amazon EC2 que executa vários Pods. Para obter mais informações, consulte Definição de preço do Amazon EC2.	Custo de instância do Amazon EC2 que executa vários Pods. Para obter mais informações, consulte Definição de preço do Amazon EC2.	Custo da memória do Fargate e das configurações da CPU. Cada Pod tem seu próprio custo. Para obter mais informações, consulte Preços do AWS Fargate.

Simplificar o ciclo de vida dos nós com grupos de nós gerenciados

Os grupos de nós gerenciados do Amazon EKS automatizam o provisionamento e o gerenciamento do ciclo de vida dos nós (instâncias do Amazon EC2) para clusters de Kubernetes do Amazon EKS.

Com os grupos de nós gerenciados do Amazon EKS, não é necessário provisionar ou registrar separadamente as instâncias do Amazon EC2 que fornecem capacidade computacional para executar as aplicações do Kubernetes. Você pode criar, atualizar ou encerrar os nós para o cluster com uma única operação. As atualizações e terminações do nó esvaziam os nós automaticamente para garantir que as aplicações continuem disponíveis.

Todos os nós gerenciados são provisionados como parte de um grupo do Amazon EC2 Auto Scaling gerenciado pelo Amazon EKS para você. Todos os recursos, inclusive as instâncias e os grupos do Auto Scaling, são executados em sua conta da AWS. Cada grupo de nós é executado em várias zonas de disponibilidade definidas por você.

Você pode adicionar um grupo de nós gerenciados a clusters novos ou existentes usando o console do Amazon EKS, o `eksctl`, a AWS CLI, a API da AWS ou infraestruturas como ferramentas de programação, incluindo o AWS CloudFormation. Os nós iniciados como parte de um grupo de nós gerenciados são etiquetados automaticamente para detecção automática pelo Kubernetes Cluster

Autoscaler. É possível usar o grupo de nós para aplicar rótulos do Kubernetes aos nós e atualizá-los a qualquer momento.

Não há custos adicionais pelo uso dos grupos de nós gerenciados pelo Amazon EKS. Você só pagará pelos recursos da AWS que provisionar. Isso inclui as instâncias do Amazon EC2, os volumes do Amazon EBS, as horas de cluster do Amazon EKS e qualquer outra infraestrutura da AWS. Não há taxas mínimas nem compromissos antecipados.

Para começar a usar um novo cluster do Amazon EKS e um grupo de nós gerenciados, consulte Conceitos básicos do Amazon EKS: AWS Management Console e AWS CLI.

Para adicionar um grupo de nós gerenciados a um cluster existente, consulte Criar um grupo de nós gerenciados para seu cluster.

Conceitos dos grupos de nós gerenciados

- Os grupos de nós gerenciados do Amazon EKS criam e gerenciam instâncias do Amazon EC2 para você.
- Todos os nós gerenciados são provisionados como parte de um grupo do Amazon EC2 Auto Scaling gerenciado pelo Amazon EKS para você. Além disso, todos os recursos, inclusive as instâncias do Amazon EC2 e os grupos do Auto Scaling, são executados em sua conta da AWS.
- O grupo do Auto Scaling de um grupo de nós gerenciados abrange todas as sub-redes que você especificar ao criar o grupo.
- O Amazon EKS adiciona etiquetas aos recursos do grupo de nós gerenciados para que eles sejam configurados para usar o Kubernetes Cluster Autoscaler.

> **Important**
>
> Se você está executando uma aplicação com estado em várias zonas de disponibilidade baseadas em volumes do Amazon EBS e usando o Kubernetes Escalar a computação em cluster com o Karpenter e o Cluster Autoscaler, deverá configurar vários grupos de nós, cada um delimitado a uma única zona de disponibilidade. Além disso, você deverá ativar o recurso `--balance-similar-node-groups`.

- Você pode usar um modelo de execução personalizado para obter um nível maior de flexibilidade e personalização ao implantar nós gerenciados. Por exemplo, é possível especificar argumentos `kubelet` extras e usar uma AMI personalizada. Para obter mais informações, consulte Personalizar nós gerenciados com modelos de execução. Se você não usar um modelo

de execução personalizado ao criar primeiro um grupo de nós gerenciados, um modelo de execução será gerado automaticamente. Não modifique manualmente esse modelo gerado automaticamente, ou ocorrerão erros.

- O Amazon EKS segue o modelo de responsabilidade compartilhada para CVEs e patches de segurança nos grupos de nós gerenciados. Quando os nós gerenciados executam uma AMI otimizada para o Amazon EKS, o Amazon EKS é responsável por criar versões da AMI com patches quando erros ou problemas forem relatados. Podemos publicar uma correção. No entanto, você é responsável por implantar essas versões de AMI com patches nos grupos de nós gerenciados. Quando os nós gerenciados executam uma AMI personalizada, você é responsável por criar versões da AMI com patches quando erros ou problemas forem relatados e, em seguida, implantar a AMI. Para obter mais informações, consulte Atualizar um grupo de nós gerenciados para seu cluster.
- Os grupos de nós gerenciados do Amazon EKS podem ser executados em sub-redes públicas e privadas. Se você executar um grupo de nós gerenciados em uma sub-rede pública depois de 22 de abril de 2020, a sub-rede deverá ter o `MapPublicIpOnLaunch` definida como true para que as instâncias possam ingressar com êxito em um cluster. Se a sub-rede pública foi criada usando `eksctl` ou os modelos do Amazon EKS fornecidos pelo AWS CloudFormation depois de 26 de março de 2020, essa configuração já estará definida como true. Se as sub-redes públicas foram criadas antes de 26 de março de 2020, será necessário alterar a configuração manualmente. Para obter mais informações, consulte Modificar o atributo de endereçamento `IPv4` público para a sua sub-rede.
- Ao implantar um grupo de nós gerenciados em sub-redes privadas, certifique-se de que ele possa acessar o Amazon ECR para extrair imagens de contêiner. Você pode fazer isso conectando um gateway NAT à tabela de rotas da sub-rede ou adicionando os seguintes endpoints de VPC do AWS PrivateLink:
  - Interface do endpoint de API do Amazon ECR - `com.amazonaws.`*`region-code`*`.ecr.api`
  - Interface do endpoint de API do registro Docker do Amazon ECR - `com.amazonaws.`*`region-code`*`.ecr.dkr`
  - Endpoint de gateway do Amazon S3 - `com.amazonaws.`*`region-code`*`.s3`

  Para conhecer alguns serviços e endpoints frequentemente, consulte Implementar clusters privados com acesso limitado à internet.
- Os grupos de nós gerenciados não podem ser implantados no AWS Outposts, no AWS Wavelength ou no AWS Local Zones.

- Você pode criar vários grupos de nós gerenciados em um único cluster. Por exemplo, é possível criar um grupo de nós com a AMI padrão do Amazon Linux otimizada para o Amazon EKS para algumas workloads e outro grupo com a variante de GPU para workloads que requerem suporte para GPU.
- Se o grupo de nós gerenciados encontrar uma falha nas verificações de status para as instâncias do Amazon EC2, o Amazon EKS retornará um código de erro para ajudar você a diagnosticar o problema. Para obter mais informações, consulte Códigos de erro para o grupo de nós gerenciados.
- O Amazon EKS adiciona rótulos do Kubernetes a instâncias de grupo de nós gerenciados. Esses rótulos fornecidos pelo Amazon EKS são prefixados com `eks.amazonaws.com`.
- O Amazon EKS drena automaticamente os nós usando a API do Kubernetes durante os encerramentos ou atualizações.
- Os orçamentos de interrupção do pod não são respeitados ao encerrar um nó com `AZRebalance` ou reduzir a contagem de nós desejada. Essas ações tentam remover o Pods do nó. Mas se demorar mais de 15 minutos, o nó será encerrado, independentemente de todos os Pods no nó serem encerrados. Para estender o período até que o nó seja encerrado, adicione um hook do ciclo de vida ao grupo do Auto Scaling. Para obter mais informações, consulte Adicionar ganchos de ciclo de vida, no Guia do usuário do Amazon EC2 Auto Scaling.
- Para executar o processo de drenagem corretamente após receber uma notificação de interrupção pontual ou uma notificação de rebalanceamento de capacidade, `CapacityRebalance` deve ser definido como `true`.
- A atualização de grupos de nós gerenciados respeita os orçamentos para interrupção do Pod definidos para os Pods. Para obter mais informações, consulte Entenda cada fase das atualizações de nós.
- Não há custos adicionais para o uso dos grupos de nós gerenciados do Amazon EKS. Você paga apenas pelo recursos da AWS que provisionar.
- Se quiser criptografar volumes do Amazon EBS para os nós, você pode implantar os nós usando um modelo de execução. Para implantar nós gerenciados com volumes criptografados do Amazon EBS sem usar um modelo de execução, criptografe todos os novos volumes do Amazon EBS criados em sua conta. Para obter mais informações, consulte Criptografia por padrão no Guia do usuário do Amazon EC2.

Tipos de capacidade do grupo de nós gerenciados

Ao criar um grupo de nós gerenciados, você pode escolher o tipo de capacidade sob demanda ou spot. O Amazon EKS implanta um grupo de nós gerenciados com um grupo do Amazon EC2 Auto Scaling que contém apenas instâncias spot sob demanda ou apenas instâncias spot do Amazon EC2. Você pode agendar Pods para aplicações tolerantes a falhas em grupos de nós gerenciados Spot e aplicações intolerantes a falhas para grupos de nós sob demanda em um único cluster do Kubernetes. Por padrão, um grupo de nós gerenciados implanta instâncias do Amazon EC2 sob demanda.

Sob demanda

Com o as instâncias sob demanda, você paga pela capacidade computacional por segundo, sem nenhum compromisso em longo prazo.

Como funciona

Por padrão, se você não especificar um tipo de capacidade, o grupo de nós gerenciados será provisionado com instâncias sob demanda. Um grupo de nós gerenciados configura um grupo do Amazon EC2 Auto Scaling em seu nome com as seguintes configurações aplicadas:

- A estratégia de alocação para provisionar capacidade sob demanda é definida como `prioritized`. O grupo de nós gerenciados usa a ordem dos tipos de instâncias transferidas na API para determinar qual tipo de instância usar primeiro para atender à capacidade sob demanda. Por exemplo, você pode especificar três tipos de instância na seguinte ordem: `c5.large`, `c4.large` e `c3.large`. Quando as instâncias sob demanda são iniciadas, o grupo de nós gerenciados atende à capacidade sob demanda, começando com o `c5.large`, depois o `c4.large` e, em seguida, o `c3.large`. Para obter mais informações, consulte Grupo do Amazon EC2 Auto Scaling) no Guia do usuário do Amazon EC2 Auto Scaling.
- O Amazon EKS adiciona o seguinte rótulo do Kubernetes a todos os nós do grupo de nós gerenciados que especifica o tipo de capacidade: `eks.amazonaws.com/capacityType: ON_DEMAND`. Você pode usar esse rótulo para agendar aplicações com estado ou intolerantes a falhas em nós sob demanda.

Spot

As instâncias spot do Amazon EC2 representam uma capacidade sobressalente do Amazon EC2 que oferecem descontos substanciais nos preços sob demanda. As instâncias spot do Amazon EC2

podem ser interrompidas com um aviso de interrupção de dois minutos, quando o EC2 precisar da capacidade de volta. Para obter mais informações, consulte Instâncias spot no Guia do Usuário do Amazon EC2. Você pode configurar um grupo de nós gerenciados com as instâncias spot do Amazon EC2 para otimizar os custos dos nós de computação em execução no cluster do Amazon EKS.

Como funciona

Para usar instâncias spot dentro de um grupo de nós gerenciados, crie o grupo definindo o tipo de capacidade como `spot`. Um grupo de nós gerenciados configura um grupo do Amazon EC2 Auto Scaling em seu nome com as seguintes práticas recomendadas Spot aplicadas:

- Para garantir que os nós spot sejam provisionados nos pools de capacidade spot ideais, a estratégia de alocação é definida como uma das seguintes opções:
  - `price-capacity-optimized` (PCO): ao criar novos grupos de nós em um cluster com o Kubernetes versão `1.28` ou superior, a estratégia de alocação é definida como `price-capacity-optimized`. No entanto, a estratégia de alocação não será alterada para grupos de nós já criados com `capacity-optimized` antes de os grupos de nós gerenciados pelo Amazon EKS começarem a oferecer suporte ao PCO.
  - `capacity-optimized` (CO): ao criar novos grupos de nós em um cluster com o Kubernetes versão `1.27` ou inferior, a estratégia de alocação é definida como `capacity-optimized`.

  Para aumentar o número de grupos de capacidade spot disponíveis para alocação de capacidade, configure um grupo de nós gerenciados para usar vários tipos de instância.
- O Rebalanceamento de capacidade spot do Amazon EC2 está habilitado para que o Amazon EKS possa drenar e reequilibrar os nós spot para minimizar a interrupção da aplicação quando um nó spot correr um risco alto de interrupção. Para obter mais informações, consulte Rebalancear a capacidade do Amazon EC2 Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.
  - Quando um nó spot recebe uma recomendação de rebalanceamento, o Amazon EKS tenta iniciar automaticamente um novo nó spot de substituição.
  - Se um aviso de interrupção do spot de dois minutos chegar antes que o nó spot de substituição esteja em um estado `Ready`, o Amazon EKS começa a drenar o nó spot que recebeu a recomendação de rebalanceamento. O Amazon EKS drena o nó da melhor forma possível. Como resultado, não há garantia de que o Amazon EKS aguardará que o nó de substituição se junte ao cluster antes de drenar o nó existente.
  - Quando um nó spot de substituição inclui o bootstrap no estado `Ready` no Kubernetes, o Amazon EKS isola e drena o nó Spot que recebeu a recomendação de rebalanceamento. Isolar

o nó spot garante que o controlador de serviço não envie novas solicitações para esse nó spot. Ele também o remove da lista de nós spot íntegros e ativos. Drenar o nó Spot garante que os Pods em execução sejam removidos normalmente.

- O Amazon EKS adiciona o seguinte rótulo do Kubernetes a todos os nós do grupo de nós gerenciados que especifica o tipo de capacidade: `eks.amazonaws.com/capacityType: SPOT`. Você pode usar esse rótulo para agendar aplicações com estado ou intolerantes a falhas em nós spot.

Considerações para selecionar um tipo de capacidade

Ao decidir se deseja implantar um grupo de nós com capacidade sob demanda ou spot, você deve considerar as seguintes condições:

- Convém usar instâncias spot para aplicações sem estado, com tolerância a falhas e flexíveis. Isso inclui workloads de treinamento em lote e de machine learning, ETLs de big data, como Apache Spark, aplicações de processamento de fila e endpoints de API sem estado. Como o spot é a capacidade sobressalente do Amazon EC2, que pode ser alterada ao longo do tempo, recomendamos usar a capacidade spot para workloads tolerantes a interrupções. Mais especificamente, a capacidade spot é adequada para workloads que podem tolerar períodos em que a capacidade necessária não está disponível.
- Recomendamos usar On-Demand para aplicações que sejam intolerantes a falhas. Isso inclui ferramentas de gerenciamento de clusters, como ferramentas operacionais e de monitoramento, implantações que exigem `StatefulSets` e aplicações com estado, como bancos de dados.
- Para maximizar a disponibilidade de suas aplicações ao usar instâncias spot, recomendamos que você configure um grupo de nós gerenciados spot para usar vários tipos de instância. Recomendamos aplicar as seguintes regras ao usar vários tipos de instância:
  - Dentro de um grupo de nós gerenciados, se você estiver usando o Cluster Autoscaler, recomendamos o uso de um conjunto flexível de tipos de instância com a mesma quantidade de vCPU e recursos de memória. Isso serve para garantir que os nós de seu cluster escalem conforme o esperado. Por exemplo, se você precisar de quatro vCPUs e oito GiB de memória, use `c3.xlarge`, `c4.xlarge`, `c5.xlarge`, `c5d.xlarge`, `c5a.xlarge`, `c5n.xlarge` ou outros tipos de instância semelhantes.
  - Para melhorar a disponibilidade das aplicações, recomendamos implantar vários grupos de nós gerenciados pelo spot. Para isso, cada grupo deverá usar um conjunto flexível de tipos de instância que tenham os mesmos recursos de vCPU e memória. Por exemplo, se você precisar de 4 vCPUs e 8 GiB de memória, recomendamos que você crie um grupo de nós gerenciados

com `c3.xlarge`, `c4.xlarge`, `c5.xlarge`, `c5d.xlarge`, `c5a.xlarge`, `c5n.xlarge` ou outros tipos de instância semelhantes, e um segundo grupo de nós gerenciados com `m3.xlarge`, `m4.xlarge`, `m5.xlarge`, `m5d.xlarge`, `m5a.xlarge`, `m5n.xlarge` ou outros tipos de instância semelhantes.

- Ao implantar o grupo de nós com o tipo de capacidade spot que estiver usando um modelo de execução personalizado, use a API para passar vários tipos de instância. Não passe um único tipo de instância pelo modelo de execução. Para obter mais informações sobre como implantar um grupo de nós usando um modelo de execução, consulte Personalizar nós gerenciados com modelos de execução.

Criar um grupo de nós gerenciados para seu cluster

Este tópico descreve como iniciar grupos de nós gerenciados do Amazon EKS com nós registrados no cluster do Amazon EKS. Depois que os nós ingressam no cluster, você pode implantar aplicações do Kubernetes neles.

Se for a primeira vez que você executa um grupo de nós gerenciados do Amazon EKS, recomendamos seguir um dos nossos guias Começar a usar o Amazon EKS. Os manuais fornecem demonstrações para criar um cluster do Amazon EKS com os nós.

⚠ Important

- Os nós do Amazon EKS são instâncias padrão do Amazon EC2. Você é cobrado com base nos preços normais do Amazon EC2. Para obter mais informações, consulte Preço do Amazon EC2.
- Não é possível criar nós gerenciados em uma Região da AWS onde você tenha o AWS Outposts, o AWS Wavelength ou o AWS Local Zones habilitado. Você pode criar nós autogerenciados em uma Região da AWS em que tenha AWS Outposts, AWS Wavelength, ou zonas locais da AWS habilitados. Para obter mais informações, consulte Criar nós autogerenciados do Amazon Linux, Criar nós Microsoft Windows autogerenciados e Criar nós Bottlerocket autogerenciados. Você também pode criar um grupo de nós autogerenciado do Amazon Linux em um Outpost. Para obter mais informações, consulte Criar nós Amazon Linux no AWS Outposts.
- Se você não especificar um ID da AMI para o arquivo `bootstrap.sh` incluso no Linux ou Bottlerocket otimizado para Amazon EKS, os grupos de nós gerenciados vão impor um número máximo no valor de `maxPods`. Para instâncias com menos de 30 vCPUs,

o número máximo é `110`. Para instâncias com mais de 30 vCPUs, o número máximo aumenta para `250`. Esses números são baseados nos limites de escalabilidade do Kubernetes e nas configurações recomendadas pelos testes internos da equipe de escalabilidade do Amazon EKS. Para obter mais informações, consulte a publicação no blog Plug-in do Amazon VPC CNI aumenta os limites de pods por nó.

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Criar um cluster do Amazon EKS..
- Um perfil do IAM existente para os nós usarem. Para criar uma, consulte Perfil do IAM em nós do Amazon EKS. Se essa função não tiver nenhuma das políticas da VPC CNI, a função separada a seguir será necessária para os pods da VPC CNI.
- (Opcional, mas recomendado) O complemento Amazon VPC CNI plugin for Kubernetes configurado com o seu próprio perfil do IAM com a política do IAM necessária anexada a ele. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).
- Familiaridade com as considerações listadas em Escolher um tipo de instância de nó do Amazon EC2 ideal. Dependendo do tipo de instância que você escolher, pode ser que haja pré-requisitos adicionais para o seu cluster e VPC.
- Para adicionar um grupo de nós gerenciados do Windows, primeiro é necessário habilitar o suporte do Windows para seu cluster. Para obter mais informações, consulte Implantar nós do Windows em clusters do EKS.

Você pode criar um grupo de nós gerenciados com o `eksctl` ou o AWS Management Console.

eksctl

Para criar um grupo de nós gerenciados com **`eksctl`**.

Este procedimento exige a versão `eksctl` `0.187.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

1. (Opcional) Se a política gerenciada do IAM, AmazonEKS_CNI_Policy, estiver anexada à Perfil do IAM em nós do Amazon EKS, recomendamos atribuí-la a um perfil do IAM que você associa à conta de serviço Kubernetes `aws-node` do Kubernetes. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).
2. Crie o grupo de nós com ou sem o uso de um modelo de execução personalizado. A especificação manual de um modelo de lançamento permite uma maior personalização de um grupo de nós. Por exemplo, pode permitir a implantação de uma AMI personalizada ou fornecer argumentos para o script de `boostrap.sh` em uma AMI otimizada do Amazon EKS. Para obter uma lista completa de todas as opções e padrões disponíveis, insira o comando a seguir.

```
eksctl create nodegroup --help
```

No comando a seguir, substitua *`my-cluster`* pelo nome do cluster e substitua *`my-mng`* pelo nome do grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.

> **Important**
>
> Se você não usar um modelo de execução personalizado ao criar um grupo de nós gerenciados, não use um superiormente para o grupo de nós. Se você não especificou um modelo de execução personalizado, o sistema gerará automaticamente um modelo de execução que não é recomendável alterar manualmente. Modificar manualmente esse modelo de execução gerado poderá causar erros automaticamente.

Sem um modelo de inicialização

O `eksctl` cria um modelo de inicialização padrão do Amazon EC2 na sua conta e implanta o grupo de nós usando um modelo de inicialização que ele criou baseado nas opções que você especifica. Antes de especificar um valor para `--node-type`, consulte Escolher um tipo de instância de nó do Amazon EC2 ideal.

Substitua *ami-family* por uma palavra-chave permitida. Para obter mais informações, consulte Setting the node AMI Family (Configurar a família de AMIs de nós) na documentação do `eksctl`. Substituir *my-key* pelo nome do par de chaves do Amazon EC2 ou da chave pública. Essa chave é usada para SSH em seus nós depois que eles forem iniciados.

> **Note**
>
> No Windows, esse comando não habilita o SSH. Em vez disso, ele associa seu par de chaves do Amazon EC2 à instância e permite que você faça RDP na instância.

Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no AWS Management Console. Para obter mais informações sobre o Linux, consulte Pares de chaves do Amazon EC2 e instâncias do Linux no Guia do usuário do Amazon EC2. Para obter mais informações sobre o Windows, consulte Pares de chaves do Amazon EC2 e instâncias do Windows no Guia do usuário do Amazon EC2.

Convém bloquear o acesso para Pod ao IMDS se as condições a seguir forem verdadeiras:

- Você planeja atribuir perfis do IAM a todas as contas de serviço do Kubernetes para que os Pods tenham apenas as permissões mínimas de que precisam.
- Nenhum dos Pods do cluster requer acesso ao serviço de metadados de instâncias do Amazon EC2 (IMDS) por outros motivos, como recuperar a Região da AWS atual.

Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Se você quiser bloquear o acesso do Pod ao IMDS, adicione a opção **`--disable-pod-imds`** ao comando a seguir.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --region region-code \
  --name my-mng \
  --node-ami-family ami-family \
  --node-type m5.large \
```

```
--nodes 3 \
--nodes-min 2 \
--nodes-max 4 \
--ssh-access \
--ssh-public-key my-key
```

As instâncias podem atribuir um número significativamente maior de endereços IP aos Pods, atribuir endereços IP aos Pods de um bloco CIDR diferente do bloco CIDR da instância e ser implantadas em um cluster sem acesso à Internet. Para obter mais informações, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2, Rede personalizada para pods, e consulte Implementar clusters privados com acesso limitado à internet para obter opções adicionais a serem incluídas no comando anterior.

Os grupos de nós gerenciados calculam e aplicam um único valor para o número máximo de Pods que podem ser executados em cada nó do grupo de nós, com base no tipo de instância. Se você criar um grupo de nós com diferentes tipos de instância, o menor valor calculado em todos os tipos de instância será aplicado como o número máximo de Pods que podem ser executados em cada tipo de instância no grupo de nós. Grupos de nós gerenciados calculam o valor usando o script referenciado em Máximo recomendado de Pods do Amazon EKS para cada tipo de instância do Amazon EC2.

Com um modelo de inicialização

O modelo de inicialização já deve existir e deve atender aos requisitos especificados em Conceitos básicos do modelo de execução.

Convém bloquear o acesso para Pod ao IMDS se as condições a seguir forem verdadeiras:

- Você planeja atribuir perfis do IAM a todas as contas de serviço do Kubernetes para que os Pods tenham apenas as permissões mínimas de que precisam.
- Nenhum dos Pods do cluster requer acesso ao serviço de metadados de instâncias do Amazon EC2 (IMDS) por outros motivos, como recuperar a Região da AWS atual.

Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Se você quiser bloquear o acesso do Pod ao IMDS, especifique as configurações necessárias no modelo de execução.

a. Copie o conteúdo a seguir para o seu dispositivo. Substitua o *example values* e, em seguida, execute o comando modificado para criar o arquivo `eks-nodegroup.yaml`. Várias configurações que você especifica ao implantar sem um modelo de execução são movidas para o modelo de execução. Se você não especificar a `version`, a versão padrão do modelo será usada.

```
cat >eks-nodegroup.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
- name: my-mng
  launchTemplate:
    id: lt-id
    version: "1"
EOF
```

Para obter uma lista completa de configurações `eksctl` do arquivo de configurações, consulte Config file schema (Esquema do arquivo de configurações) na documentação do `eksctl` As instâncias podem, opcionalmente, atribuir um número significativamente maior de endereços IP aos Pods, atribuir endereços IP a Pods de um bloco CIDR diferente do bloco CIDR da instância, usar o runtime do `containerd` e ser implantadas em um cluster sem acesso externo à Internet. Para obter mais informações, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2, Rede personalizada para pods e Teste a migração do Amazon Linux 2 de Docker para `containerd` e consulte Implementar clusters privados com acesso limitado à internet para obter opções adicionais a serem adicionadas ao arquivo de configurações.

Se você não especificou um ID de AMI no modelo de lançamento, os grupos de nós gerenciados calculam e aplicam um único valor para o número máximo de Pods que podem ser executados em cada nó do grupo de nós, com base no tipo de instância. Se você criar um grupo de nós com diferentes tipos de instância, o menor valor calculado em todos os tipos de instância será aplicado como o número máximo de Pods que podem ser executados em cada tipo de instância no grupo de nós. Grupos de nós

gerenciados calculam o valor usando o script referenciado em Máximo recomendado de Pods do Amazon EKS para cada tipo de instância do Amazon EC2.

Se você especificou um ID de AMI no modelo de inicialização, especifique o número máximo de Pods que podem ser executados em cada nó do grupo de nós, se você estiver usando redes personalizadas ou se quiser aumentar o número de endereços IP atribuídos à instância. Para obter mais informações, consulte Máximo recomendado de Pods do Amazon EKS para cada tipo de instância do Amazon EC2.

b. Implante o grupo de nós com o comando a seguir.

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

AWS Management Console

Para criar um grupo de nós gerenciados usando o AWS Management Console

1. Aguarde até que o status do cluster seja exibido como `ACTIVE`. Não é possível criar um grupo de nós gerenciados para um cluster que ainda não esteja `ACTIVE`.
2. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
3. Escolha o nome do cluster em que você deseja criar um grupo de nós gerenciados.
4. Selecione a guia Compute (Computação).
5. Escolha Add Node Group (Adicionar grupo de nós).
6. Na página Configure node group (Configurar o grupo de nós) preencha os parâmetros adequadamente e escolha Next (Próximo).

   - Name (Nome) – Insira um nome exclusivo para o grupo de nós gerenciados. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.
   - Node IAM role (Perfil do IAM do nó): escolha a função da instância do nó a ser usada com o grupo de nós. Para obter mais informações, consulte Perfil do IAM em nós do Amazon EKS.

   > **Important**
   >
   > - Não é possível usar o mesmo perfil usado para criar clusters.

- Recomendamos usar uma função que não esteja em uso atualmente por um grupo de nós autogerenciados. Caso contrário, planeje usar com um novo grupo de nós autogerenciados. Para obter mais informações, consulte Excluir um grupo de nós gerenciados do seu cluster.

- Usar: (opcional) escolha se deseja usar um modelo de execução existente. Selecione um Nome de modelo de inicialização. Depois, selecione Launch template version (Versão do modelo de inicialização). Se você não selecionar uma versão, o Amazon EKS usará a versão padrão do modelo. Os modelos de inicialização permitem uma maior personalização do grupo de nós, por exemplo, permitem que você implante uma AMI personalizada, atribua um número significativamente maior de endereços IP aos Pods, atribua endereços IP a Pods de um bloco CIDR diferente do bloco da instância, habilite o runtime do `containerd` para as instâncias, e também que implante nós em um cluster sem acesso de saída à Internet. Para obter mais informações, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2, Rede personalizada para pods, Teste a migração do Amazon Linux 2 de Docker para `containerd` e Implementar clusters privados com acesso limitado à internet.

  O modelo de execução deve cumprir os requisitos do Personalizar nós gerenciados com modelos de execução. Se você não usar seu próprio modelo de execução, a API do Amazon EKS criará um modelo de execução padrão do Amazon EC2 em sua conta e implantará o grupo de nós usando o modelo de execução padrão.

  Se você implementar Perfis do IAM para contas de serviço, atribua as permissões necessárias diretamente a todos os Pod que precisem ter acesso aos serviços da AWS, e nenhum Pods no cluster exigirá acesso ao IMDS por outros motivos, como recuperar a Região da AWS atual, depois, você também poderá desabilitar o acesso ao IMDS para Pods que não usam redes de host em um modelo de inicialização. Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.
- Rótulos do Kubernetes: (Opcional) Você pode optar por aplicar rótulos do Kubernetes aos nós no grupo de nós gerenciados.
- Kubernetes taints: (Opcional) É possível aplicar Kubernetes taints aos nós do seu grupo de nós gerenciados. As opções disponíveis naAs opções disponíveis no menu Effect (Efeito) são **`NoSchedule`**, **`NoExecute`** e **`PreferNoSchedule`**. Para obter mais informações, consulte Evitar que Pods seja agendado em nós específicos.

- Etiquetas – (Opcional) você pode optar por aplicar etiquetas ao grupo de nós gerenciados pelo Amazon EKS. Essas etiquetas não se propagam para outros recursos no grupo de nós, como instâncias ou grupos do Auto Scaling. Para obter mais informações, consulte Organizar recursos do Amazon EKS com tags.

7. Na página Set compute and scaling configuration (Definir configuração de escalabilidade e computação), preencha os parâmetros adequadamente e escolha Next (Próximo).

   - AMI type (Tipo de AMI): selecione um tipo de AMI. Se estiver implantando instâncias Arm, não deixe de revisar as considerações em AMIs Amazon Linux Arm otimizadas para Amazon EKS antes de implantar..

     Se você especificou um modelo de inicialização na página anterior e especificou nele uma AMI, não poderá selecionar um valor. O valor do modelo é exibido. A AMI especificada no modelo deve atender aos requisitos em Especificar uma AMI.

   - Tipo de capacidade – Selecione um tipo de capacidade. Para obter mais informações sobre a escolha de um tipo de capacidade, consulte Tipos de capacidade do grupo de nós gerenciados. Não é possível misturar diferentes tipos de capacidade dentro do mesmo grupo de nós. Se você quiser usar os dois tipos de capacidade, crie grupos de nós separados, cada um com seus próprios tipos de capacidade e instância.

   - Tipos de instância: por padrão, um ou mais tipos de instância são especificados. Para remover um tipo de instância padrão, selecione a caixa de seleção X no lado direito do tipo de instância. Escolha o tipo de instância a ser usado no grupo de nós gerenciados. Para obter mais informações, consulte Escolher um tipo de instância de nó do Amazon EC2 ideal.

     O console exibe um conjunto de tipos de instância comumente usados. Se você precisar criar um grupo de nós gerenciados com um tipo de instância que não é exibido, use `eksctl`, a AWS CLI, o AWS CloudFormation ou um SDK para criar o grupo de nós. Se você especificou um modelo de execução na página anterior, não será possível selecionar um valor, porque o tipo de instância deve ser especificado no modelo de execução. O valor do modelo de execução é exibido. Se você selecionou Spot para o Tipo de capacidade, recomendamos especificar vários tipos de instância para melhorar a disponibilidade.

   - Tamanho do disco – Insira o tamanho do disco (em GiB) a ser usado para o volume raiz do nó.

     Se você especificou um modelo de execução na página anterior, não será possível selecionar um valor, porque ele deverá ser especificado no modelo de execução.

- Tamanho desejado – Especifique o número atual de nós que o grupo de nós gerenciados deverá manter na inicialização.

> **Note**
>
> O Amazon EKS não reduz nem aumenta automaticamente a escala na horizontal do grupo de nós. No entanto, você pode configurar o Kubernetes Cluster Autoscaler para fazer isso por você.

- Tamanho mínimo – Especifique o número mínimo de nós para o qual o grupo de nós gerenciados pode ser reduzido.
- Tamanho máximo – Especifique o número máximo de nós para o qual o grupo de nós gerenciados pode ser expandido.
- Configuração da atualização de grupo de nós – (Opcional) Você pode selecionar o número ou porcentagem de nós a serem atualizados em paralelo. Esses nós estarão indisponíveis durante a atualização. Para o Máximo disponível, selecione uma das seguintes opções e especifique um Valor:
  - Number (Número): selecione e especifique o número de nós em seu grupo de nós que podem ser atualizados em paralelo.
  - Percentage (Porcentagem): selecione e especifique a porcentagem de nós em seu grupo de nós que podem ser atualizados em paralelo. Isso será útil se você tiver um grande número de nós em seu grupo de nós.

8. Na página Specify networking (Especificar a rede), preencha os parâmetros conforme necessário e, em seguida, escolha Next (Próximo).

- Sub-redes – Escolha as sub-redes nas quais iniciar os nós gerenciados.

> **Important**
>
> Se você está executando uma aplicação com estado em várias zonas de disponibilidade baseadas em volumes do Amazon EBS e usando o Kubernetes Escalar a computação em cluster com o Karpenter e o Cluster Autoscaler, deverá configurar vários grupos de nós, cada um delimitado a uma única zona de disponibilidade. Além disso, você deverá ativar o recurso `--balance-similar-node-groups`.

> **Important**
>
> - Se você escolher uma sub-rede pública e o cluster tiver somente o endpoint do servidor de API público habilitado, a sub-rede deverá ter o `MapPublicIPOnLaunch` definido como `true` para que as instâncias ingressem com êxito em um cluster. Se a sub-rede foi criada usando `eksctl` ou os modelos do Amazon EKS fornecidos pelo AWS CloudFormation depois de 26 de março de 2020, essa configuração já estará definida como `true`. Se as sub-redes foram criadas com modelos do `eksctl` ou AWS CloudFormation antes de 26 de março de 2020, será necessário alterar a configuração manualmente. Para obter mais informações, consulte Modificar o atributo de endereçamento `IPv4` público para a sua sub-rede.
> - Se você usar um modelo de execução e especificar várias interfaces de rede, o Amazon EC2 não atribuirá automaticamente um endereço `IPv4` público, mesmo que o `MapPublicIpOnLaunch` seja definido como `true`. Para que os nós ingressem no cluster nesse cenário, você deve habilitar o endpoint do servidor de API privado do cluster ou iniciar nós em uma sub-rede privada com acesso de saída à Internet fornecido por meio de um método alternativo, como um Gateway NAT. Para obter mais informações, consulte Endereçamento de IP de instâncias do Amazon EC2, no Guia do Usuário do Amazon EC2.

- Configurar o acesso SSH aos nós (opcional). Habilitar o SSH permite que você se conecte às suas instâncias e reúna informações de diagnóstico quando houver problemas. É altamente recomendado habilitar o acesso remoto ao criar o grupo de nós. Não será possível habilitar o acesso remoto depois que o grupo estiver criado.

  Se você optou por usar um modelo de execução, essa opção não será exibida. Para habilitar o acesso remoto aos nós, especifique um par de chaves no modelo de execução e verifique se a porta adequada está aberta aos nós nos grupos de segurança especificados no modelo de execução. Para obter mais informações, consulte Usar grupos de segurança personalizados.

> **Note**
>
> No Windows, esse comando não habilita o SSH. Em vez disso, ele associa seu par de chaves do Amazon EC2 à instância e permite que você faça RDP na instância.

- Para o par de chaves SSH (Opcional), escolha uma chave SSH do Amazon EC2 para usar. Para obter mais informações sobre o Linux, consulte Pares de chaves do Amazon EC2 e instâncias do Linux no Guia do usuário do Amazon EC2. Para obter mais informações sobre o Windows, consulte Pares de chaves do Amazon EC2 e instâncias do Windows no Guia do usuário do Amazon EC2. Se você optar por usar um modelo de execução, não será possível selecionar um. Quando uma chave SSH do Amazon EC2 é fornecida para grupos de nós que usam AMIs Bottlerocket, o contêiner administrativo também é habilitado. Para obter mais informações, consulte Admin container(Contêiner administrador) no GitHub.
- Para Allow SSH remote access from (Permitir o acesso remoto de SSH) se você quiser limitar o acesso a instâncias específicas, selecione os grupos de segurança associados a essas instâncias. Se você não selecionar grupos de segurança específicos, o acesso SSH será permitido de qualquer lugar na Internet (`0.0.0.0/0`).

9. Na página Review and create (Revisar e criar), reveja a configuração do grupo de nós gerenciados e escolha Create (Criar).

   Se os nós não conseguirem se juntar ao cluster, consulte Worker nodes fail to join cluster (Falha nos nós ao ingressar no cluster) no manual de solução de problemas.

10. Observe o status de seus nós e aguarde até que eles atinjam o status `Ready`.

    ```
    kubectl get nodes --watch
    ```

11. (Somente nós de GPU) Se tiver escolhido um tipo de instância de GPU e a AMI acelerada otimizada para o Amazon EKS, você deverá aplicar o Plug-in de dispositivo NVIDIA para Kubernetes como um DaemonSet no cluster. Substitua *vX.X.X* pela versão desejada de NVIDIA/k8s-device-plugin antes de executar o comando a seguir.

    ```
    kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/deployments/static/nvidia-device-plugin.yml
    ```

Agora que você tem um cluster do Amazon EKS com nós, já poderá iniciar a instalação dos complementos do Kubernetes e implantar as aplicações no cluster. Os seguintes tópicos de documentação ajudam a estender a funcionalidade do seu cluster:

- A entidade principal do IAM que criou o cluster é a única entidade que pode fazer chamadas ao servidor da API do Kubernetes usando `kubectl` ou o AWS Management Console. Se quiser que outras entidades principais do IAM tenham acesso ao cluster, será necessário adicioná-los. Para ter mais informações, consulte Conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes e Permissões obrigatórias.
- Convém bloquear o acesso para Pod ao IMDS se as condições a seguir forem verdadeiras:
  - Você planeja atribuir perfis do IAM a todas as contas de serviço do Kubernetes para que os Pods tenham apenas as permissões mínimas de que precisam.
  - Nenhum dos Pods do cluster requer acesso ao serviço de metadados de instâncias do Amazon EC2 (IMDS) por outros motivos, como recuperar a Região da AWS atual.

  Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.
- Escalar a computação em cluster com o Karpenter e o Cluster Autoscaler: configure o Kubernetes Cluster Autoscaler para ajustar automaticamente o número de nós nos grupos de nós.
- Implante uma aplicação de exemplo no cluster.
- Organizar e monitorar recursos de clusters – Saiba como usar ferramentas importantes para gerenciar o cluster.

Criar um grupo de nós gerenciados com blocos de capacidade para ML

Os blocos de capacidade para machine learning (ML) permitem que você reserve instâncias com GPU para uma data futura, a fim de dar suporte a workloads de ML de curta duração. Para obter mais informações, consulte Blocos de capacidade para ML no Guia do Usuário do Amazon EC2 para Instâncias Linux.

Considerações

> **Important**
>
> - Os blocos de capacidade só estão disponíveis para determinados tipos de instância do Amazon EC2 e Regiões da AWS. Para obter informações sobre compatibilidade, consulte

Trabalhar com pré-requisitos de blocos de capacidade no Guia do usuário do Amazon EC2 para instâncias Linux.

- Para obter mais informações, consulte Usar blocos de capacidade para workloads de machine learning no Guia do usuário do Amazon EC2 Auto Scaling.
- Grupos de nós gerenciados com blocos de capacidade só podem ser criados com modelos de lançamento personalizados.
- Ao atualizar grupos de nós gerenciados com blocos de capacidade, certifique-se de que o tamanho desejado do grupo de nós esteja definido como `0`.

Criar um grupo de nós gerenciados com blocos de capacidade do Amazon EC2

Você pode usar blocos de capacidade com grupos de nós gerenciados do Amazon EKS para provisionar e escalar os nós de processamento acelerados por GPU. Os exemplos de modelos do AWS CloudFormation a seguir não abrangem todos os aspectos necessários em clusters de produção. Normalmente, você também vai querer um script de bootstrapping para unir o nó ao cluster e especificar a AMI acelerada do Amazon EKS. Para obter mais informações, consulte Criar um grupo de nós gerenciados para seu cluster.

1. Crie um modelo de execução apropriado para as workloads e que funcione com grupos de nós gerenciados do Amazon EKS. Para obter mais informações, consulte Personalizar nós gerenciados com modelos de execução.

   Além dos requisitos dos procedimentos acima, certifique-se de que `LaunchTemplateData` inclua o seguinte:

   - `InstanceMarketOptions` com `MarketType` definido como `"capacity-block"`
   - `CapacityReservationSpecification: CapacityReservationTarget` com `CapacityReservationId` definido como bloco de capacidade (por exemplo:`cr-02168da1478b509e0`)
   - `InstanceType` definido como um tipo de instância compatível com blocos de capacidade (por exemplo: *`p5.48xlarge`*)

   Veja a seguir um trecho de um modelo do CloudFormation para criar um modelo de execução direcionado a um bloco de capacidade. Para criar um grupo personalizado de nós gerenciados da AMI, você também pode adicionar os parâmetros `ImageId` e `UserData`.

```
NodeLaunchTemplate:
  Type: "AWS::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      InstanceType: p5.48xlarge
```

2. Use o modelo de execução para criar um grupo de nós gerenciados.

   Veja a seguir um exemplo de comando para criar grupos de nós para blocos de capacidade. Substitua *example-values* pelos aplicáveis ao seu cluster.

   Ao criar o grupo de nós gerenciados do bloco de capacidade, faça o seguinte:

   - Defina `capacity-type` como `"CAPACITY_BLOCK"` . Se o tipo de capacidade não estiver definido como `"CAPACITY_BLOCK"`, ou se algum dos outros valores obrigatórios do modelo de execução acima estiver ausente, a solicitação de criação será rejeitada.
   - Ao especificar `subnets` na solicitação de criação, certifique-se de especificar somente a sub-rede na mesma zona de disponibilidade como a reserva de capacidade.
   - Se você especificar um `desiredSize` diferente de zero na solicitação de criação, o Amazon EKS atenderá a essa especificação ao criar o grupo do Auto Scaling (ASG). No entanto, se a solicitação de criação for feita antes que a reserva de capacidade esteja ativa, o ASG não poderá executar instâncias do Amazon EC2 até que elas se tornem ativas. Como resultado, as atividades de escalabilidade do ASG terão erros de execução. Sempre que a reserva se tornar ativa, a execução das instâncias terá êxito e o ASG terá a escala aumentada verticalmente para o `desiredSize` mencionado no momento da criação.

```
aws eks create-nodegroup \
    --cluster-name my-cluster \
    --nodegroup-name my-mng \
    --node-role node-role-arn \
    --region region-code \
    --subnets subnet-id \
    --scaling-config minSize=node-group-min-size,maxSize=node-group-max-
size,desiredSize=node-group-desired-size \
```

```
    --capacity-type "CAPACITY_BLOCK" \
    --launch-template id="lt-id",version=1
```

3. Verifique se os nós se unem após o aumento da escala verticalmente. Os clusters do Amazon EKS que usam grupos de nós gerenciados com blocos de capacidade não realizam nenhuma validação de que as instâncias executadas realmente se juntam e se registram no cluster.
4. Caso defina `desiredSize` como `0` no momento da criação, você terá opções diferentes para aumentar a escala verticalmente do grupo de nós quando a reserva de capacidade ficar ativa:
   - Crie uma política de escalabilidade programada para o ASG que se alinhe ao horário de início da reserva do bloco de capacidade. Para obter mais informações, consulte Escalabilidade agendada para o Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.
   - Use o console do Amazon EKS ou `eks update-nodegroup-config` para atualizar a configuração de escalabilidade e defina o tamanho desejado do grupo de nós.
   - Use o Autoscaler do cluster do Kubernetes. Para obter mais informações, consulte Autoscaler do cluster na AWS.
5. O grupo de nós agora está pronto para workloads e Pods para ser programado.
6. Para que os Pods sejam facilmente drenados antes do término da reserva, o Amazon EKS usa uma política de escalabilidade programada para reduzir a escala verticalmente do tamanho do grupo de nós para `0`. Essa escalabilidade programada será denominada `Amazon EKS Node Group Capacity Scaledown Before Reservation End`. Recomendamos não editar ou excluir essa ação.

   O Amazon EC2 começa a encerrar as instâncias 30 minutos antes do horário de término da reserva. Como resultado, o Amazon EKS vai configurar uma redução vertical da escala programada no grupo de nós 40 minutos antes do final da reserva, a fim de remover Pods com segurança e facilidade.

Atualizar um grupo de nós gerenciados para seu cluster

Quando você inicia uma atualização do grupo de nós gerenciados, o Amazon EKS atualiza automaticamente os nós para você, concluindo as etapas listadas em Entenda cada fase das atualizações de nós. Se você estiver usando uma AMI otimizada do Amazon EKS, o Amazon EKS aplica automaticamente os patches de segurança mais recentes e as atualizações do sistema operacional aos nós como parte da versão mais recente da AMI.

Existem vários cenários nos quais é útil atualizar a versão ou a configuração do grupo de nós gerenciados do Amazon EKS:

- Você atualizou a versão do Kubernetes para seu cluster do Amazon EKS e quer atualizar os nós para usar a mesma versão do Kubernetes.
- Uma nova versão da AMI está disponível para o grupo de nós gerenciados. Para obter mais informações sobre as versões da AMI, consulte estas seções:
  - Recuperar informações da versão da AMI do Amazon Linux
  - Criar nós com AMIs do Bottlerocket otimizadas
  - Recuperar informações da versão da AMI do Windows
- Você deseja ajustar a contagem mínima, máxima ou desejada das instâncias no seu grupo de nós gerenciados.
- Você deseja adicionar ou remover os rótulos do Kubernetes das instâncias no seu grupo de nós gerenciados.
- Você deseja adicionar ou remover as tags da AWS do seu grupo de nós gerenciados.
- Você precisa implantar uma nova versão de um modelo de execução com alterações de configuração, como uma AMI personalizada atualizada.
- Você implantou a versão `1.9.0` ou superior do complemento Amazon VPC CNI, habilitou o complemento para delegação de prefixos e deseja ter novas Instâncias do AWSNitro System em um grupo de nós para compatibilidade com um número significativamente maior de Pods. Para obter mais informações, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2.
- Você habilitou a delegação de prefixo IP para nós do Windows e deseja que novas instâncias do AWS Nitro System em um grupo de nós ofereçam suporte a um número significativamente maior de Pods. Para obter mais informações, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2.

Se houver uma versão da AMI mais recente do que a versão do Kubernetes do grupo de nós gerenciado, você poderá atualizá-la para usar essa nova versão da AMI. Da mesma forma, se o cluster estiver executando uma versão mais recente do Kubernetes do que o grupo de nós, você poderá atualizar o grupo de nós para usar a versão mais recente da AMI correspondente à versão do Kubernetes do cluster.

Quando um nó em um grupo de nós gerenciados é encerrado devido a uma operação de escalabilidade ou atualização, os Pods nesse nó são antes drenados. Para obter mais informações, consulte Entenda cada fase das atualizações de nós.

Atualizar uma versão do grupo de nós

Para atualizar uma versão do grupo de nós com o `eksctl` e o AWS Management Console. A versão para a qual você atualiza não pode ser superior à versão do ambiente de gerenciamento.

eksctl

Para atualizar uma versão do grupo de nós com **`eksctl`**

- Atualize um grupo de nós gerenciados para a versão mais recente da AMI da mesma versão do Kubernetes atualmente implantada nos nós com o comando a seguir. Substitua *`example value`* por seus próprios valores.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code
```

> **Note**
>
> Se você estiver atualizando um grupo de nós implantado com um modelo de execução para uma nova versão do modelo de execução, adicione o `--launch-template-version` *`version-number`* ao comando anterior. O modelo de execução deve cumprir os requisitos descritos em Personalizar nós gerenciados com modelos de execução. Se o modelo de execução incluir uma AMI personalizada, a AMI deverá cumprir os requisitos em Especificar uma AMI. Quando você atualiza o grupo de nós para uma versão mais recente do modelo de execução, todos os nós são reciclados para corresponder à nova configuração da versão do modelo de execução especificado.
>
> Não é possível atualizar diretamente um grupo de nós implantado sem um modelo de execução para uma nova versão do modelo de execução. Em vez disso, você deve implantar um novo grupo de nós usando o modelo de execução para atualizar o grupo de nós para uma nova versão de modelo de execução.

Você pode atualizar um grupo de nós para a mesma versão que a versão do Kubernetes do ambiente de gerenciamento. Por exemplo, se tiver um cluster executando o Kubernetes `1.29`, você poderá atualizar os nós que estão executando o Kubernetes `1.28` para a versão `1.29` com o comando a seguir.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code \
  --kubernetes-version=1.29
```

AWS Management Console

Para atualizar uma versão do grupo de nós com o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha o cluster que contém o grupo de nós a ser atualizado.
3. Se pelo menos um grupo de nós tiver uma atualização disponível, uma caixa será exibida na parte superior da página notificando você sobre a atualização disponível. Se você selecionar a guia Compute (Computação), verá a opção Update now (Atualizar agora) na coluna AMI release version (Versão de AMI) na tabela Node Groups (Grupos de nós) para o grupo de nós que tem atualizações disponíveis. Para atualizar o grupo de nós, selecione Update now (Atualizar agora).

   Você não verá uma notificação para grupos de nós que foram implantados com uma AMI personalizada. Se os nós forem implantados com uma AMI personalizada, conclua as etapas a seguir para implantar uma nova AMI personalizada atualizada.

   a. Crie uma nova versão da AMI.

   b. Crie uma nova versão do modelo de execução com o novo ID da AMI.

   c. Atualize os nós para a nova versão do modelo de execução.
4. Na caixa de diálogo Update node group version (Atualizar versão do grupo de nós), ative ou desative as seguintes opções:

- Update Node Group version (Atualizar versão do grupo de nós): essa opção não estará disponível se você tiver implantado uma AMI personalizada ou se a AMI otimizada do Amazon EKS estiver atualmente na versão mais recente do cluster.
- Change launch template version (Alterar versão do modelo de inicialização): essa opção não estará disponível se o grupo de nós for implantado sem um modelo de inicialização personalizado. Você só pode atualizar a versão do modelo de execução para um grupo de nós que tenha sido implantado com um modelo de execução personalizado. Selecione a Launch template version (Versão do modelo de inicialização) para a qual você deseja atualizar o grupo de nós. Se o grupo de nós estiver configurado com uma AMI personalizada, a versão selecionada também deverá especificar uma AMI. Quando você atualiza para uma versão mais recente do modelo de execução, todos os nós são reciclados para corresponder à nova configuração da versão do modelo de execução especificada.

5. Em Update strategy (Estratégia de atualização), selecione uma das seguintes opções:

   - Rolling update (Atualização contínua): essa opção respeita os orçamentos de interrupções de Pod no cluster. As atualizações falharão se houver um problema de orçamento de interrupção de Pod que impeça que o Amazon EKS drene corretamente os Pods que estão em execução no grupo de nós.
   - Force update (Forçar atualização): essa opção não respeita os orçamentos de interrupção de Pod. As atualizações ocorrem a despeito dos problemas de orçamento de interrupção de Pod, forçando a reinicialização do nó.

6. Selecione Atualizar.

Editar uma configuração do grupo de nós

Você pode modificar algumas configurações de um grupo de nós gerenciados.

Para editar uma configuração do grupo de nós

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha o cluster que contém o grupo de nós a ser editado.
3. Selecione a guia Compute (Computação).
4. Selecione o grupo de nós a ser editado e escolha Edit (Editar).
5. (Opcional) Na página Edit node group (Editar grupo de nós) faça o seguinte:

a. Edite a configuração de escalonamento do grupo de nós.

   - Tamanho desejado – Especifique o número atual de nós que o grupo de nós gerenciados deve manter.
   - Tamanho mínimo – Especifique o número mínimo de nós para o qual o grupo de nós gerenciados pode ser reduzido.
   - Tamanho máximo – Especifique o número máximo de nós para o qual o grupo de nós gerenciados pode ser expandido. Para obter o número máximo de nós permitidos em um grupo de nós, consulte Visualizar e gerenciar o Amazon EKS e as cotas de serviço do Fargate.

b. (Opcional) Adicione ou remova rótulos do Kubernetes dos nós do grupo de nós. Os rótulos mostrados aqui são apenas aqueles que você aplicou com o Amazon EKS. Outros rótulos que não são exibidos aqui podem existir nos nós.

c. (Opcional) Adicione ou remova taints do Kubernetes dos nós do grupo de nós. As taints adicionadas podem ter o efeito de **`NoSchedule`**, **`NoExecute`** ou **`PreferNoSchedule`**. Para obter mais informações, consulte Evitar que Pods seja agendado em nós específicos.

d. (Opcional) Adicione ou remova tags do recurso de grupo de nós. Essas etiquetas são aplicadas somente ao grupo de nós do Amazon EKS. Não são propagadas em outros recursos como sub-redes ou instâncias do Amazon EC2 no grupo de nós.

e. (Opcional) Edite a configuração da atualização do grupo de nós. Selecione o número ou a porcentagem.

   - Number (Número): selecione e especifique o número de nós em seu grupo de nós que podem ser atualizados em paralelo. Esses nós estarão indisponíveis durante a atualização.
   - Percentage (Porcentagem): selecione e especifique a porcentagem de nós em seu grupo de nós que podem ser atualizados em paralelo. Esses nós estarão indisponíveis durante a atualização. Isso será útil se houver muitos nós em seu grupo de nós.

f. Quando terminar de editar, escolha Save changes (Salvar alterações).

Entenda cada fase das atualizações de nós

A estratégia de atualização do nó de processamento gerenciado do Amazon EKS tem quatro fases diferentes descritas nas seções a seguir.

Fase de configuração

A fase de configuração contém estas etapas:

1. Ele cria uma nova versão do modelo de execução do Amazon EC2 para o grupo do Auto Scaling que estiver associado ao grupo de nós. A nova versão do modelo de inicialização usa a AMI de destino ou uma versão do modelo de inicialização personalizado para a atualização.
2. Ele atualiza o grupo do Auto Scaling para usar a versão mais recente do modelo de execução.
3. Ele determina a quantidade máxima de nós a serem atualizados em paralelo usando a propriedade `updateConfig` do grupo de nós. O máximo indisponível tem uma cota de 100 nós. O valor padrão é um nó. Para obter mais informações, consulte a propriedade `updateConfig` na Referência da API do Amazon EKS.

Fase de aumento de escala vertical

Na atualização de nós em um grupo de nós gerenciados, os nós atualizados são iniciados na mesma zona de disponibilidade daqueles que estão sendo atualizados. Para garantir esse posicionamento, usamos o rebalanceamento de zonas de disponibilidade do Amazon EC2. Para obter mais informações, consulte Rebalanceamento de zonas de disponibilidade no Guia do usuário do Amazon EC2 Auto Scaling. Para atender a esse requisito, é possível a inicialização de até duas instâncias por zona de disponibilidade no grupo de nós gerenciados.

A fase de aumento de escala vertical tem estas etapas:

1. Ele aumenta o tamanho máximo do grupo do Auto Scaling e o tamanho desejado pelo que for maior:
   - Até uma a duas vezes o número de zonas de disponibilidade na região em que o grupo do Auto Scaling está implantado.
   - O máximo indisponível de atualização.

     Por exemplo, se o grupo de nós tiver cinco zonas de disponibilidade e `maxUnavailable` como um, o processo de atualização poderá iniciar no máximo dez nós. No entanto, se `maxUnavailable` for 20 (ou qualquer valor superior a 10), o processo iniciaria 20 novos nós.
2. Depois de escalar o grupo do Auto Scaling, ele verifica se os nós que usam a configuração mais recente estão presentes no grupo de nós. Essa etapa só é concluída corretamente quando atende a estes critérios:
   - Pelo menos um novo nó é iniciado em todas as zonas de disponibilidade em que o nó existe.

- Cada novo nó deve estar no estado `Ready`.
- Os novos nós devem ter rótulos aplicados ao Amazon EKS.

  São os rótulos do Amazon EKS aplicados aos nós de processamento em um grupo de nós regulares:

  - `eks.amazonaws.com/nodegroup-image=$amiName`
  - `eks.amazonaws.com/nodegroup=$nodeGroupName`

  São os rótulos do Amazon EKS aplicados aos nós de processamento em um modelo de execução personalizado ou em um grupo de nós da AMI:

  - `eks.amazonaws.com/nodegroup-image=$amiName`
  - `eks.amazonaws.com/nodegroup=$nodeGroupName`
  - `eks.amazonaws.com/sourceLaunchTemplateId=$launchTemplateId`
  - `eks.amazonaws.com/sourceLaunchTemplateVersion=$launchTemplateVersion`

3. Ele marca os nós como não programáveis para evitar o agendamento de novos Pods. Ele também rotula os nós `node.kubernetes.io/exclude-from-external-load-balancers=true` para remover os nós dos balanceadores de carga antes de encerrar os nós.

A seguir, estão os motivos conhecidos que causam um erro `NodeCreationFailure` nesta fase:

Capacidade insuficiente na zona de disponibilidade

Existe a possibilidade de que a zona de disponibilidade não tenha capacidade dos tipos de instância solicitados. É recomendável configurar vários tipos de instância ao criar um grupo de nós gerenciados.

Limites de instâncias do EC2 na sua conta

Pode ser necessário aumentar o número de instâncias do Amazon EC2 que a conta pode executar simultaneamente usando o Service Quotas. Para obter mais informações, consulte EC2 Service Quotas no Guia do usuário do Amazon Elastic Compute Cloud para instâncias do Linux.

Dados de usuário personalizados

Os dados de usuário personalizados às vezes podem interromper o processo de bootstrap. Esse cenário pode fazer com que o `kubelet` não seja iniciado no nó ou com que os nós não recebam os rótulos esperados do Amazon EKS. Para obter mais informações, consulte Especificar uma AMI.

Quaisquer alterações que prejudiquem a integridade ou a prontidão de um nó

Pressão no disco do nó, pressão na memória e condições semelhantes podem impedir que um nó passe para o estado `Ready`.

Fase de atualização

A fase de atualização contém estas etapas:

1. Ele seleciona aleatoriamente um nó que precisa ser atualizado, até o máximo indisponível configurado para o grupo de nós.
2. Ele drena os Pods do nó. Se os Pods não saírem do nó em 15 minutos e não houver um sinalizador de força, a fase de atualização falhará com um erro `PodEvictionFailure`. Nesse cenário, você pode aplicar o sinalizador de força com a solicitação de `update-nodegroup-version` para excluir os Pods.
3. Ele isola o nó após cada Pod ser removido e aguarda 60 segundos. Isso é feito para que o controlador de serviço não envie novas solicitações para este nó e remova esse nó da lista de nós ativos.
4. Ele envia uma solicitação de encerramento para o grupo do Auto Scaling do nó isolado.
5. Ele repete as etapas de atualização anteriores até que não haja nós no grupo de nós implantados com a versão anterior do modelo de execução.

A seguir, estão os motivos conhecidos que causam um erro `PodEvictionFailure` nesta fase:

PDB agressivo

Um PDB agressivo está definido no Pod ou há vários PDBs apontando para o mesmo Pod.

Implantação que tolera todas as taints

Depois que todo Pod é removido, espera-se que o nó fique vazio porque o nó sofreu taints nas etapas anteriores. Porém, se a implantação tolerar todos os taints, é mais provável que o nó não esteja vazio, causando falha na remoção do Pod.

Fase de redução de escala vertical

A fase de redução de escala vertical diminui o tamanho máximo do grupo do Auto Scaling e o tamanho desejado para retornar aos valores anteriores ao início da atualização.

Se o fluxo de trabalho Upgrade determinar que o Cluster Autoscaler está aumentando a escala na vertical do grupo de nós durante a fase de redução de escala na vertical do fluxo de trabalho, ele será encerrado imediatamente sem trazer o grupo de nós de volta ao tamanho original.

Evitar que Pods seja agendado em nós específicos

O Amazon EKS oferece suporte à configuração de taints do Kubernetes por meio dos grupos de nós gerenciados. Taints e as tolerâncias trabalham em conjunto para garantir que os Pods não sejam agendados em nós inapropriados. Uma ou mais taints podem ser aplicadas a um nó. Isso indica que o nó não deve aceitar Pods que não tolerem taints. As tolerâncias são aplicadas aos Pods e permitem, mas não exigem, que os Pods sejam agendados em nós com taints correspondentes. Para obter mais informações, consulte Taints e Tolerâncias na documentação do Kubernetes.

Taints de nós do Kubernetes podem ser aplicadas a grupos de nós gerenciados novos e existentes usando o AWS Management Console ou por meio da API do Amazon EKS.

- Para obter informações sobre como criar um grupo de nós com um taint usando o AWS Management Console, consulte Criar um grupo de nós gerenciados para seu cluster.
- Veja a seguir um exemplo de criação de um grupo de nós com uma taint, usando a AWS CLI:

```
aws eks create-nodegroup \
 --cli-input-json '
{
  "clusterName": "my-cluster",
  "nodegroupName": "node-taints-example",
  "subnets": [
     "subnet-1234567890abcdef0",
     "subnet-abcdef01234567890",
     "subnet-021345abcdef67890"
   ],
  "nodeRole": "arn:aws:iam::111122223333:role/AmazonEKSNodeRole",
  "taints": [
     {
         "key": "dedicated",
         "value": "gpuGroup",
         "effect": "NO_SCHEDULE"
     }
   ]
}'
```

Para obter mais informações e exemplos de uso, consulte taint na documentação de referência do Kubernetes.

> **Note**
>
> - Os taints podem ser atualizados após a criação do grupo de nós usando a API `UpdateNodegroupConfig`.
> - A chave da taint deve começar com uma letra ou um número. Pode conter letras, números, hifens (`-`), pontos (`.`) e sublinhados (`_`). Pode ter até 63 caracteres.
> - Opcionalmente, a chave da taint pode começar com um prefixo de subdomínio DNS e um `/` único. Se começar com um prefixo de subdomínio DNS, poderá ter 253 caracteres.
> - O valor é opcional e deve começar com uma letra ou um número. Pode conter letras, números, hifens (`-`), pontos (`.`) e sublinhados (`_`). Pode ter até 63 caracteres.
> - Ao usar o Kubernetes diretamente ou o AWS Management Console, o efeito do taint deve ser **`NoSchedule`**, **`PreferNoSchedule`** ou **`NoExecute`**. No entanto, ao usar a AWS CLI ou API, o efeito de taint deve ser **`NO_SCHEDULE`**, **`PREFER_NO_SCHEDULE`** ou **`NO_EXECUTE`**.
> - Um máximo de 50 taints são permitidos por grupo de nós.
> - Se os taints criados usando um grupo de nós gerenciados forem removidos manualmente de um nó, o Amazon EKS não os adicionará de volta ao nó. Isso será válido mesmo se as contaminações forem especificadas na configuração do grupo de nós gerenciados.

Você pode usar o comando `aws eks update-nodegroup-config` da AWS CLI para adicionar, remover ou substituir taints em grupos de nós gerenciados.

Personalizar nós gerenciados com modelos de execução

Para obter o nível mais alto de personalização, é possível implementar nós gerenciados usando seu próprio modelo de execução. O uso de um modelo de execução permite recursos como os seguintes:

- Fornecer argumentos de inicialização na implantação de um nó, como argumentos de `kubelet` extras.
- Atribuir endereços IP a Pods de um bloco CIDR diferente do endereço IP atribuído ao nó.
- Implantar sua própria AMI personalizada nos nós.
- Implantar sua própria CNI personalizada nos nós.

Ao fornecer seu próprio modelo de execução na primeira criação de um grupo de nós gerenciados, você também terá maior flexibilidade posteriormente. Contanto que você implemente um grupo de nós gerenciados com seu próprio modelo de execução, é possível atualizá-lo iterativamente para uma versão diferente do mesmo modelo de execução. Quando você atualiza para o grupo de nós para uma versão mais recente do modelo de execução, todos os nós do grupo são reciclados para corresponder à nova configuração da versão do modelo de execução especificada.

Os grupos de nós gerenciados são sempre implantados com um modelo de execução do grupo do Amazon EC2 Auto Scaling. Quando você não fornece um modelo de execução, a API do Amazon EKS cria um modelo com valores padrão em sua conta automaticamente. No entanto, não recomendamos modificar os modelos de execução gerados automaticamente. Além disso, os grupos de nós existentes que não usarem um modelo de execução personalizado não poderão ser atualizados diretamente. Em vez disso, você deverá criar um novo grupo de nós com um modelo de execução personalizado para fazer isso.

Conceitos básicos do modelo de execução

É possível criar um modelo de execução do Amazon EC2 Auto Scaling com o AWS Management Console, o AWS CLI ou um AWS SDK. Para obter mais informações, consulte Criação de um modelo de execução para um grupo do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling. Algumas das configurações em um modelo de execução são semelhantes às usadas para a configuração do nó gerenciado. Ao implantar ou atualizar um grupo de nós com um modelo de execução, algumas configurações devem ser especificadas na configuração do grupo de nós ou no modelo de execução. Não especifique uma configuração em ambos os locais. Se uma configuração existir onde não deveria, as operações, como criar ou atualizar um grupo de nós, falharão.

A tabela a seguir lista as configurações proibidas em um modelo de execução. Também são listadas configurações semelhantes, se houver, que são obrigatórias na configuração do grupo de nós gerenciados. As configurações listadas são as configurações que aparecem no console. Elas podem ter nomes semelhantes, mas diferentes na AWS CLI e no SDK.

Modelo de execução – Proibido	Configuração do grupo de nós do Amazon EKS
Subnet (Sub-rede) em Network interfaces (Interfaces de rede) (Adicionar interface de rede)	Subnet (Sub-redes) em Node Group network configuration (Configuração de rede do grupo de nós) na página Specify networking (Especificar rede)

Modelo de execução – Proibido	Configuração do grupo de nós do Amazon EKS
IAM instance profile (Perfil de instância do IAM) em Advanced details (Detalhes avançados)	Node IAM Role (Função do IAM do nó) em Node Group configuration (Configuração do grupo de nós) na página Configure Node Group (Configurar o grupo de nós)
Shutdown behavior (Comportamento de desligamento) e Stop - Hibernate behavior (Parar - comportamento de hibernação) em Advanced details (Detalhes avançados). Reter o padrão Não incluir na configuração do modelo de execução para as duas configura ções.	Não há equivalente. O Amazon EKS deve controlar o ciclo de vida da instância, não o grupo Auto Scaling.

A tabela a seguir lista as configurações proibidas em uma configuração de grupo de nós gerenciados. Também lista configurações semelhantes, se houver, que serão necessárias em um modelo de execução. As configurações listadas são as configurações que aparecem no console. Elas podem ter nomes semelhantes na AWS CLI e no SDK.

Configuração do grupo de nós do Amazon EKS - Proibidas	Modelo de execução
(Somente se você especificou uma AMI personalizada em um modelo de inicializ ação) AMI type (Tipo de AMI) em Node Group compute configuration (Configuração da computação do grupo de nós) na página Set compute and scaling configuration (Definir configuração de computação e escalabilidade): o console exibe Specified in launch template (Especificado no modelo de inicialização) e o ID da AMI que foi especificado. Se Application and OS Images (Amazon Machine Image) [Imagens de aplicações e sistemas operacionais (imagem de máquina	Application and OS Images (Amazon Machine Image) [Imagens de aplicações e sistemas operacionais (imagem de máquina da Amazon)] em Launch template contents (Conteúdo do modelo de execução): especifiq ue um ID se você tiver um dos seguintes requisitos: • Usar uma AMI personalizada. Se você especificar uma AMI que não atende aos requisitos listados em Especificar uma AMI, haverá falha na implantação do grupo de nós.

Configuração do grupo de nós do Amazon EKS - Proibidas	Modelo de execução
da Amazon)] não tiver sido especificado no modelo de execução, você poderá selecionar uma AMI na configuração do grupo de nós.	• Deseja fornecer dados de usuário para fornecer argumentos para o arquivo `bootstrap.sh` incluído com uma AMI otimizada do Amazon EKS. É possível habilitar que suas instâncias atribuam um número significativamente maior de endereços IP aos Pods, atribuam endereços IP aos Pods de um bloco de CIDR diferente do endereço da instância ou implantem um cluster privado sem acesso de saída à Internet. Para obter mais informações, consulte os tópicos a seguir. • Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2 • Rede personalizada para pods • Implementar clusters privados com acesso limitado à internet • Especificar uma AMI
Tamanho do disco em Node Group compute configuration (Configuração de computação do grupo de nós) a página Set compute and scaling configuration (Definir a configuração da computação e escalabilidade). Exibe o console Especificado no modelo de execução.	Size (Tamanho) em Storage (Volumes) Armazenamento (Volumes) (Add New volume) (Adicionar novo volume). Você deve especificar isso no modelo de execução.
Par de chaves SSH em Node Group configura tion (Configuração do grupo de nós) na página Specify Networking (Especificar rede) — O console exibe a chave especificada no modelo de execução ou exibe Not specified in launch template (Não especificado no modelo de execução).	Key pair name (Nome do par de chaves) em Key pair (login) (Par de chaves).

Configuração do grupo de nós do Amazon EKS - Proibidas	Modelo de execução
Não é possível especificar grupos de segurança de origem que tenham permissão de acesso remoto ao usar um modelo de execução.	Security groups (Grupos de segurança) em Network settings (Configurações de rede) para a instância ou Security groups (Grupos de segurança) em Network interfaces (Interfac es de rede) (Add network interface (Adicionar interface de rede), mas não os dois. Para obter mais informações, consulte Usar grupos de segurança personalizados.

Note

- Se você implantar um grupo de nós usando um modelo de execução, especifique zero ou um tipo de instância em Launch template contents (Iniciar o conteúdo do modelo) em um modelo de execução. Se preferir, você pode especificar de 0 a 20 tipos de instância em Instance types (Tipos de instância) na página Set compute and scaling configuration (Definir configuração de computação e escalabilidade) no console. Ou você pode fazer isso com outras ferramentas que utilizam a API do Amazon EKS. Se você especificar um tipo de instância em um modelo de execução e usar esse modelo para implantar o grupo de nós, não será possível especificar nenhum tipo de instância no console ou usar outras ferramentas que usam a API do Amazon EKS. Se você não especificar um tipo de instância em um modelo de execução, no console ou usando outras ferramentas que usem a API do Amazon EKS, será usado o tipo de instância `t3.medium`. Se o grupo de nós estiver usando o tipo de capacidade spot, recomendamos especificar vários tipos de instância usando o console. Para obter mais informações, consulte Tipos de capacidade do grupo de nós gerenciados.
- Se algum contêiner implantado no grupo de nós usar o Serviço de Metadados de Instância Versão 2, defina a propriedade Metadata response hop limit (Limite de salto de resposta) como 2 no modelo de execução. Para obter mais informações, consulte Metadados da instância e dados do usuário no Manual do usuário do Amazon EC2. Se você implantar um grupo de nós gerenciados sem usar um modelo de execução personalizado, esse valor será definido automaticamente para o grupo de nós no modelo de execução padrão.

Etiquetar instâncias do Amazon EC2

Você pode usar o parâmetro `TagSpecification` de um modelo de execução para especificar quais etiquetas serão aplicadas às instâncias do Amazon EC2 em seu grupo de nós. A entidade do IAM que chama o método das APIs `CreateNodegroup` ou `UpdateNodegroupVersion` devem ter permissões para `ec2:RunInstances` e `ec2:CreateTags`, e as etiquetas devem ser adicionadas ao modelo de execução.

Usar grupos de segurança personalizados

Você pode usar um modelo de execução para especificar os grupos de segurança do Amazon EC2 a serem aplicados às instâncias do grupo de nós. Isso pode ser no parâmetro dos grupos de segurança do nível de instância ou como parte dos parâmetros de configuração da interface de rede. Porém, não é possível criar um modelo de execução que especifique o nível da instância e os grupos de segurança da interface de rede. Considere as seguintes condições que se aplicam ao uso de grupos de segurança personalizados com grupos de nós gerenciados:

- Quando o AWS Management Console é usado, o Amazon EKS permite modelos de execução somente com uma única especificação de interface de rede.
- Por padrão, o Amazon EKS aplica o Grupo de segurança de clusterpara as instâncias no grupo de nós, a fim de facilitar a comunicação entre os nós e o plano de controle. Se você especificar grupos de segurança personalizados no modelo de execução usando uma das opções mencionadas anteriormente, o Amazon EKS não adicionará o grupo de segurança do cluster. Então, você deve garantir que as regras de entrada e saída dos grupos de segurança permitam a comunicação com o endpoint do cluster. Se as regras do grupo de segurança estiverem incorretas, os nós de processamento não poderão ingressar no cluster. Para obter mais informações sobre regras do grupo de segurança, consulte Considerações e requisitos sobre grupos de segurança do Amazon EKS.
- Se você precisar de acesso SSH às instâncias no grupo de nós, inclua um grupo de segurança que permita esse acesso.

Dados do usuário do Amazon EC2

O modelo de execução contém uma seção para dados de usuário personalizados. É possível especificar configurações para o grupo de nós desta seção sem criar AMIs personalizadas individuais manualmente. Para obter mais informações sobre as configurações disponíveis para o Bottlerocket, consulte Using user data (Usar dados de usuário) no GitHub.

Você pode fornecer dados de usuário do Amazon EC2 no modelo de execução usando `cloud-init` na inicialização das instâncias. Para obter mais informações, consulte a documentação de cloud-init. Os dados de usuário podem ser usados para executar operações de configuração comuns. Isso inclui as seguintes operações:

- Incluindo usuários ou grupos
- Instalar pacotes

Os dados de usuários do Amazon EC2 em modelo de execução que são usados com grupos de nós gerenciados devem estar no formato arquivo MIME de várias partes para AMIs do Amazon Linux e em formato TOML para AMIs do Bottlerocket. Isso ocorre porque seus dados de usuário são mesclados com os dados de usuário do Amazon EKS necessários para que os nós participem do cluster. Não especifique nenhum comando nos dados de usuário que iniciem ou modifiquem `kubelet`. Isso é executado como parte dos dados de usuário mesclados pelo Amazon EKS. Certos parâmetros do `kubelet`, como a definição de rótulos em nós, podem ser configurados diretamente por meio da API de grupos de nós gerenciados.

> **Note**
>
> Para obter mais informações sobre personalização avançada do `kubelet`, inclusive para iniciá-la manualmente ou passar parâmetros de configuração personalizados, consulte Especificar uma AMI. Seu um ID de AMI personalizada for especificado em um modelo de execução, o Amazon EKS não mesclará os dados de usuário.

Os detalhes a seguir fornecem mais informações sobre a seção de dados de usuário.

Amazon Linux 2 user data

Você pode combinar vários blocos de dados de usuário em um único arquivo MIME de várias partes. Por exemplo, você pode combinar um boothook de nuvem que configure o daemon do Docker com um script do shell de dados do usuário que instale um pacote personalizado. Um arquivo em várias partes MIME consiste nos seguintes componentes:

- O tipo de conteúdo e a declaração de limite da parte: `Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="`
- A declaração da versão MIME: `MIME-Version: 1.0`
- Um ou mais blocos de dados do usuário, que contêm os seguintes componentes:

- O limite de abertura, que sinaliza o início de um bloco de dados do usuário: `--==MYBOUNDARY==`
- A declaração de tipo de conteúdo para o bloco: `Content-Type: text/cloud-config; charset="us-ascii"`. Para obter mais informações sobre os tipos de conteúdo, consulte a documentação do cloud-init.
- O conteúdo de dados de usuário (por exemplo, uma lista de comandos de shell ou diretivas `cloud-init`).
- O limite de fechamento, que sinaliza o término do arquivo em várias partes MIME: `--==MYBOUNDARY==--`

Veja a seguir um exemplo de arquivo MIME com várias partes que você pode usar para criar seu próprio.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
echo "Running custom user data script"

--==MYBOUNDARY==--
```

Amazon Linux 2023 user data

O Amazon Linux 2023 (AL2023) introduz um novo processo de inicialização do nó `nodeadm` que usa um esquema de configuração YAML. Se estiver usando grupos de nós autogerenciados ou uma AMI com um modelo de inicialização, será necessário fornecer, de forma explícita, metadados de cluster adicionais ao criar um novo grupo de nós. Veja a seguir um exemplo dos parâmetros mínimos necessários, em que `apiServerEndpoint`, `certificateAuthority` e `cidr` do serviço passaram a ser necessários:

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
```

```
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

Normalmente, você definirá essa configuração nos dados do usuário no estado em que se encontra ou incorporada em um documento MIME com várias partes:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig spec: [...]

--BOUNDARY--
```

No AL2, os metadados desses parâmetros eram revelados na chamada de API `DescribeCluster` do Amazon EKS. Com o AL2023, esse comportamento foi alterado, uma vez que a chamada de API adicional corre o risco de sofrer controle de utilização durante grandes aumentos de escala vertical para nós. Essa alteração não afetará você se estiver usando grupos de nós gerenciados sem um modelo de inicialização ou se estiver usando o Karpenter. Para obter mais informações sobre `certificateAuthority` e `cidr` do serviço, consulte `DescribeCluster` na Referência de APIs do Amazon EKS.

Bottlerocket user data

O Bottlerocket estrutura dados de usuário no formato TOML. Informe os dados de usuário a serem mesclados com os dados de usuário fornecidos pelo Amazon EKS. Por exemplo, você pode fornecer outras configurações do `kubelet`.

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

Para obter mais informações sobre as configurações compatíveis, consulte a documentação Bottlerocket. É possível configurar rótulos de nós e taints em seus dados de usuário. Porém,

recomendamos configurá-los em seu grupo de nós. O Amazon EKS aplica essas configurações quando você faz isso.

Quando os dados de usuário são mesclados, a formatação não é preservada, mas o conteúdo permanece o mesmo. A configuração que você fornece nos dados de usuário substitui todas as configurações definidas pelo Amazon EKS. Dessa maneira, se você definir `settings.kubernetes.max-pods` ou `settings.kubernetes.cluster-dns-ip`, esses valores dos dados de usuário são aplicados aos nós.

O Amazon EKS não oferece suporte a todos os TOML válidos. Veja a seguir uma lista de formatos conhecidos não compatíveis:

- Aspas em chaves entre aspas: `'quoted "value"' = "value"`
- Aspas de escape em valores: `str = "I'm a string. \"You can quote me\""`
- Floats mistos e números inteiros: `numbers = [ 0.1, 0.2, 0.5, 1, 2, 5 ]`
- Tipos mistos em matrizes: `contributors = ["foo@example.com", { name = "Baz", email = "baz@example.com" }]`
- Cabeçalhos entre colchetes com chaves entre aspas: `[foo."bar.baz"]`

Windows user data

Os dados de usuário do Windows usam comandos do PowerShell. Ao criar um grupo de nós gerenciados, os dados de usuário personalizados se combinam com os dados de usuário gerenciados do Amazon EKS. Os comandos do PowerShell vêm em primeiro lugar, seguidos pelos comandos de dados de usuário gerenciados, tudo dentro de uma tag **`<powershell></powershell>`**.

> **Note**
>
> Quando nenhum ID de AMI for especificado no modelo de execução, não use o script de boostrap do Amazon EKS para Windows nos dados de usuário para configurar o Amazon EKS.

A seguir, um exemplo de dados do usuário.

```
<powershell>
```

```
Write-Host "Running custom user data script"
</powershell>
```

Especificar uma AMI

Se você tiver um dos seguintes requisitos, especifique um ID de AMI na caixa `ImageId` do modelo de execução. Selecione o requisito que você tem para obter informações adicionais.

Fornecer dados de usuário para passar argumentos para o arquivo **`bootstrap.sh`** incluído com uma AMI do Linux/Bottlerocket otimizada para o Amazon EKS

Bootstrapping é um termo utilizado para descrever o processo de adicionar comandos que podem ser executados quando uma instância é iniciada. Por exemplo, a inicialização permite o uso de argumentos `kubelet` extras. Você pode passar os argumentos para o script `bootstrap.sh` usando `eksctl` sem especificar um modelo de inicialização. Ou faça isso especificando as informações na seção de dados de usuário de um modelo de execução.

eksctl without specifying a launch template

Crie um arquivo denominado *`my-nodegroup.yaml`* com o seguinte conteúdo: Substitua *`example value`* por seus próprios valores. Os argumentos `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` são opcionais. No entanto, defini-los permite que o script `bootstrap.sh` evite fazer uma chamada `describeCluster`. Isso é útil em configurações de cluster privado ou clusters em que há aumento e redução de escala na horizontal com frequência. Para obter mais informações sobre o script `bootstrap.sh`, consulte o arquivo `bootstrap.sh` no GitHub.

- O único argumento necessário é o nome do cluster (*`my-cluster`*).
- Para recuperar um ID de AMI otimizado para `ami-`*`1234567890abcdef0`*, você pode usar as tabelas nas seguintes seções:
  - Recuperar IDs de AMI do Amazon Linux recomendadas
  - Recuperar IDs de AMI do Bottlerocket recomendadas
  - Recuperar IDs de AMI do Microsoft Windows recomendadas
- Para recuperar o *`certificate-authority`* do seu cluster, execute o comando a seguir.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Para recuperar o *api-server-endpoint* do seu cluster, execute o comando a seguir.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster --region region-code
```

- O valor para o `--dns-cluster-ip` é o serviço CIDR com `.10` no final. Para recuperar o *service-cidr* do seu cluster, execute o comando a seguir. Por exemplo, se o valor retornado for `ipv4 10.100.0.0/16`, então seu valor será *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Este exemplo fornece um argumento adicional do `kubelet` para definir um valor `max-pods` personalizado usando o valor do script `bootstrap.sh` incluído com a AMI otimizada para o Amazon EKS. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Para obter ajuda com a seleção de *my-max-pods-value*, consulte Máximo recomendado de Pods do Amazon EKS para cada tipo de instância do Amazon EC2.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    instanceType: m5.large
    privateNetworking: true
    disableIMDSv1: true
    labels: { x86-al2-specified-mng }
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster \
        --b64-cluster-ca certificate-authority \
        --apiserver-endpoint api-server-endpoint \
        --dns-cluster-ip service-cidr.10 \
        --kubelet-extra-args '--max-pods=my-max-pods-value' \
```

```
        --use-max-pods false
```

Para cada opção de arquivo `eksctl config` disponível, consulte Esquema do arquivo config, na documentação do `eksctl`. O utilitário `eksctl` ainda cria um modelo de inicialização para você e preenche seus dados de usuário com dados fornecidos no arquivo `config`.

Crie um grupo de nós com o comando a seguir.

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

User data in a launch template

Especifique as seguintes informações na seção de dados do usuário do modelo de execução. Substitua cada *example value* por seus próprios valores. Os argumentos `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` são opcionais. No entanto, defini-los permite que o script `bootstrap.sh` evite fazer uma chamada `describeCluster`. Isso é útil em configurações de cluster privado ou clusters em que há aumento e redução de escala na horizontal com frequência. Para obter mais informações sobre o script `bootstrap.sh`, consulte o arquivo `bootstrap.sh` no GitHub.

- O único argumento necessário é o nome do cluster (*my-cluster*).
- Para recuperar o *certificate-authority* do seu cluster, execute o comando a seguir.

  ```
  aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
   --name my-cluster --region region-code
  ```

- Para recuperar o *api-server-endpoint* do seu cluster, execute o comando a seguir.

  ```
  aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
  cluster --region region-code
  ```

- O valor para o `--dns-cluster-ip` é o serviço CIDR com `.10` no final. Para recuperar o *service-cidr* do seu cluster, execute o comando a seguir. Por exemplo, se o valor retornado for `ipv4 10.100.0.0/16`, então seu valor será *10.100.0.10*.

  ```
  aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
   --output text --name my-cluster --region region-code
  ```

- Este exemplo fornece um argumento adicional do `kubelet` para definir um valor `max-pods` personalizado usando o valor do script `bootstrap.sh` incluído com a AMI otimizada para o

Amazon EKS. Para obter ajuda com a seleção de *my-max-pods-value*, consulte Máximo recomendado de Pods do Amazon EKS para cada tipo de instância do Amazon EC2.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
/etc/eks/bootstrap.sh my-cluster \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
  --use-max-pods false

--==MYBOUNDARY==--
```

Fornecer dados de usuário para passar argumentos para o arquivo **`Start-EKSBootstrap.ps1`** incluído com uma AMI do Windows otimizada do Amazon EKS

Bootstrapping é um termo utilizado para descrever o processo de adicionar comandos que podem ser executados quando uma instância é iniciada. Você pode passar os argumentos para o script `Start-EKSBootstrap.ps1` usando `eksctl` sem especificar um modelo de inicialização. Ou faça isso especificando as informações na seção de dados de usuário de um modelo de execução.

Se você quiser especificar uma ID de AMI do Windows, tenha em mente as seguintes considerações:

- Você deve usar um modelo de inicialização e fornecer os comandos de bootstrap necessários na seção de dados do usuário. Para recuperar o ID do Windows desejado, você pode usar a tabela em Criar nós com AMIs do Windows otimizadas.
- Existem vários limites e condições. Por exemplo, você deve adicionar `eks:kube-proxy-windows` ao mapa de configuração do AWS IAM Authenticator. Para obter mais informações, consulte Limites e condições ao especificar uma ID de AMI.

Especifique as seguintes informações na seção de dados do usuário do modelo de execução. Substitua cada *example value* por seus próprios valores. Os argumentos `-APIServerEndpoint`, `-Base64ClusterCA` e `-DNSClusterIP` são opcionais. No entanto, defini-los permite que o script `Start-EKSBootstrap.ps1` evite fazer uma chamada `describeCluster`.

- O único argumento necessário é o nome do cluster (*my-cluster*).
- Para recuperar o *certificate-authority* do seu cluster, execute o comando a seguir.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- Para recuperar o *api-server-endpoint* do seu cluster, execute o comando a seguir.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- O valor para o `--dns-cluster-ip` é o serviço CIDR com `.10` no final. Para recuperar o *service-cidr* do seu cluster, execute o comando a seguir. Por exemplo, se o valor retornado for `ipv4 10.100.0.0/16`, então seu valor será *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- Para obter argumentos adicionais, consulte Parâmetros de configuração do script de bootstrap.

> **Note**
>
> Se você estiver usando um CIDR de serviço personalizado, será necessário especificá-lo usando o parâmetro `-ServiceCIDR`. Caso contrário, a resolução de DNS Pods no cluster falhará.

```
<powershell>
[string]$EKSBootstrapScriptFile = "$env:ProgramFiles\Amazon\EKS\Start-EKSBootstrap.ps1"
& $EKSBootstrapScriptFile -EKSClusterName my-cluster `
  -Base64ClusterCA certificate-authority `
  -APIServerEndpoint api-server-endpoint `
  -DNSClusterIP service-cidr.10
</powershell>
```

Executar uma AMI personalizada devido a requisitos específicos de segurança, conformidade ou política interna

Para obter mais informações, consulte Imagens de máquina da Amazon (AMIs) no Guia do usuário do Amazon EC2. A especificação de compilação da AMI do Amazon EKS contém recursos e scripts de configuração para desenvolver uma AMI do Amazon EKS personalizada baseada no Amazon Linux. Para obter mais informações, consulte Amazon EKS AMI Build Specification (Especificação de criação de AMI do Amazon EKS) no GitHub. Para criar AMIs personalizadas instaladas com outros sistemas operacionais, consulte Amazon EKS Sample Custom AMIs (Amostras de AMIs personalizadas do Amazon EKS) no GitHub.

> **Important**
>
> Ao especificar uma AMI, o Amazon EKS não mescla nenhum dado do usuário. Em vez disso, você é responsável por fornecer os comandos do `bootstrap` para que os nós se integrem ao cluster. Se os nós não conseguirem se integrar ao cluster, as ações do Amazon EKS `CreateNodegroup` e do `UpdateNodegroupVersion` também falharão.

Limites e condições ao especificar uma ID de AMI

Estes são os limites e condições envolvidos na especificação de um ID de AMI com grupos de nós gerenciados:

- Você deve criar um novo grupo de nós para alternar entre especificar um ID de AMI em um modelo de execução e não especificar um ID de AMI.
- Você não é notificado no console quando uma versão mais recente da AMI estiver disponível. Para atualizar seu grupo de nós para uma versão mais recente da AMI, é necessário criar uma nova versão do modelo de execução com um ID de AMI atualizado. Em seguida, é necessário atualizar o grupo de nós com a nova versão do modelo de execução.
- Os campos a seguir não podem ser definidos na API se você especificar um ID de AMI:
  - `amiType`
  - `releaseVersion`
  - `version`
- Qualquer `taints` conjunto na API é aplicado de forma assíncrona se você especificar um ID de AMI. Para aplicar contaminações antes de um nó ingressar no cluster, você deve transmiti-las para

`kubelet` os dados do usuário usando a sinalização da linha de comando `--register-with-taints`. Para obter mais informações, consulte a `kubelet` documentação do Kubernetes.

- Ao especificar uma ID de AMI personalizada para grupos de nós gerenciados do Windows, adicione `eks:kube-proxy-windows` ao mapa de configuração do AWS IAM Authenticator. Essa API é necessária para o DNS funcionar bem.

  1. Abra o mapa de configuração do AWS IAM Authenticator para edição.

     ```
     kubectl edit -n kube-system cm aws-auth
     ```

  2. Adicione essa entrada à lista de `groups` abaixo de cada `rolearn` associado aos nós do Windows. O mapa de configuração deve ser semelhante ao `aws-auth-cm-windows.yaml`.

     ```
     - eks:kube-proxy-windows
     ```

  3. Salve o arquivo e saia do seu editor de texto.

Excluir um grupo de nós gerenciados do seu cluster

Este tópico descreve como você pode excluir um grupo de nós gerenciados do Amazon EKS. Quando você exclui um grupo de nós gerenciados, o Amazon EKS primeiro define o tamanho mínimo, máximo e desejado do grupo do Auto Scaling como zero. Isso faz com que seu grupo de nós reduza a escala na vertical.

Antes de cada instância ser encerrada, o Amazon EKS envia um sinal para drenar os Pods daquele nó. Se os Pods não forem drenados depois de alguns minutos, o Amazon EKS permitirá que o Auto Scaling continue o encerramento da instância. Depois que todas as instâncias forem terminadas, o grupo do Auto Scaling será excluído.

> **Important**
>
> Se você excluir um grupo de nós gerenciado que usa um perfil do IAM do nó que não é usado por outro grupo de nós gerenciados no cluster, a função será removida do `ConfigMap aws-auth`. Se algum grupo de nós autogerenciados no cluster estiver usando a mesma função do IAM do nó, os nós autogerenciados serão movidos para o status `NotReady`. Além disso, a operação do cluster também é interrompida. Para adicionar um mapeamento para a função que você está usando somente para os grupos de nós autogerenciados, consulte Criar entradas de acesso, caso a versão da plataforma do seu

cluster seja pelo menos a versão mínima listada na seção de pré-requisitos de Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS. Se a versão da plataforma for anterior à versão mínima exigida para entradas de acesso, você poderá adicionar a entrada novamente ao `ConfigMap` `aws-auth`. Para obter mais informações, insira `eksctl create iamidentitymapping --help` no seu terminal.

Você pode excluir um grupo de nós gerenciados com o `eksctl` ou o AWS Management Console.

eksctl

Para excluir um grupo de nós gerenciados com **`eksctl`**

Insira o comando da a seguir. Substitua *`example value`* por seus próprios valores.

```
eksctl delete nodegroup \
  --cluster my-cluster \
  --name my-mng \
  --region region-code
```

Para obter mais opções, consulte Excluir e drenar grupos de nós na documentação do `eksctl`.

AWS Management Console

Para excluir o grupo de nós gerenciados usando o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Na página Clusters, escolha o cluster que contém o grupo de nós a ser excluído.
3. Na página do cluster selecionado, escolha a guia Computação.
4. Na seção Node groups (Grupos de nós), escolha o grupo de nós a ser excluído. Em seguida, selecione Excluir.
5. Na caixa de diálogo de confirmação Excluir grupo de nós, insira o nome do grupo de nós. Em seguida, selecione Excluir.

AWS CLI

Para excluir o grupo de nós gerenciados usando o AWS CLI

1. Insira o comando da a seguir. Substitua *`example value`* por seus próprios valores.

```
aws eks delete-nodegroup \
  --cluster-name my-cluster \
  --nodegroup-name my-mng \
  --region region-code
```

2. Use as teclas de seta do teclado para percorrer a saída da resposta. Pressione a tecla **q** quando terminar.

Para obter mais opções, consulte o comando `delete-nodegroup` na Referência de comandos da AWS CLI.

Fazer a manutenção dos nós por conta própria com nós autogerenciados

Um cluster contém um ou mais nós do Amazon EC2 nos quais os Pods estão agendados. Os nós do Amazon EKS são executados na conta da AWS e se conectam ao ambiente de gerenciamento do cluster pelo endpoint do servidor da API do cluster. Você é cobrado por elas com base nos preços do Amazon EC2. Para obter mais informações, consulte Definição de preço do Amazon EC2.

Um cluster pode conter vários grupos de nós. Cada grupo de nós contém um ou mais nós implementados em um Amazon EC2 Auto Scalinggrupo. O tipo de instância dos nós dentro do grupo pode variar, como ao usar a seleção de tipo de instância baseada em atributos com Karpenter. Todas as instâncias de um grupo de nós devem usar a perfil do IAM de nós do Amazon EKS.

O Amazon EKS fornece imagens de máquina da Amazon (AMI) especializada que são chamadas de AMIs otimizadas do Amazon EKS. As AMIs estão configuradas para funcionar com o Amazon EKS. Seus componentes incluem `containerd`, `kubelet` e o AWS IAM Authenticator. As AMIs também contêm um script de bootstrap especializado que permite que ele descubra e conecte-se ao plano de controle do cluster automaticamente.

Se você restringir o acesso ao endpoint público do cluster usando blocos CIDR, é recomendável também habilitar o acesso ao endpoint privado. Isso serve para que os nós possam se comunicar com o cluster. Sem o endpoint privado ativado, os blocos CIDR especificados para acesso público devem incluir as origens de saída da VPC. Para obter mais informações, consulte Controlar o acesso à rede ao endpoint do servidor de API do cluster.

Para adicionar nós autogerenciados ao cluster do Amazon EKS, consulte os tópicos a seguir. Se você iniciar os nós autogerenciados manualmente, adicione a etiqueta a seguir a cada nó. Para obter mais informações, consulte Adicionar e excluir tags em um recurso individual. Se você seguir as etapas nos guias a seguir, a etiqueta necessária será adicionada ao nó para você.

Chave	Valor
`kubernetes.io/cluster/ my-cluster`	`owned`

Para obter mais informações sobre nós, de um ponto de vista geral do Kubernetes, consulte Nodes (Nós) na documentação do Kubernetes.

Tópicos

- Criar nós autogerenciados do Amazon Linux
- Criar nós Bottlerocket autogerenciados
- Criar nós Microsoft Windows autogerenciados
- Criar nós Ubuntu Linux autogerenciados
- Atualizar nós autogerenciados para seu cluster

Criar nós autogerenciados do Amazon Linux

Este tópico descreve como você pode iniciar grupos do Auto Scaling de nós do Linux que são registrados no cluster do Amazon EKS. Depois que os nós ingressam no cluster, você pode implantar aplicações do Kubernetes neles. Além disso, é possível iniciar nós autogerenciados do Amazon Linux ao usar `eksctl` ou o AWS Management Console. Se você precisar iniciar nós no AWS Outposts, consulte Criar nós Amazon Linux no AWS Outposts.

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Criar um cluster do Amazon EKS.. Se você tiver sub-redes na Região da AWS com o AWS Outposts, o AWS Wavelength ou o AWS Local Zones habilitado, tais sub-redes não devem ter sido aprovadas na criação do seu cluster.
- Um perfil do IAM existente para os nós usarem. Para criar uma, consulte Perfil do IAM em nós do Amazon EKS. Se essa função não tiver nenhuma das políticas da VPC CNI, a função separada a seguir será necessária para os pods da VPC CNI.

- (Opcional, mas recomendado) O complemento Amazon VPC CNI plugin for Kubernetes configurado com o seu próprio perfil do IAM com a política do IAM necessária anexada a ele. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).
- Familiaridade com as considerações listadas em Escolher um tipo de instância de nó do Amazon EC2 ideal. Dependendo do tipo de instância que você escolher, pode ser que haja pré-requisitos adicionais para o seu cluster e VPC.

eksctl

> **Note**
>
> No momento, a ferramenta `eksctl` não é compatível com o Amazon Linux 2023.

Pré-requisito

Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Para iniciar nós autogerenciados do Linux usando **`eksctl`**

1. (Opcional) Se a política gerenciada do IAM, AmazonEKS_CNI_Policy, estiver anexada à Perfil do IAM em nós do Amazon EKS, recomendamos atribuí-la a um perfil do IAM que você associa à conta de serviço Kubernetes `aws-node` do Kubernetes. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).
2. O comando a seguir cria um grupo de nós em um cluster existente. Substitua *`al-nodes`* por um nome para o seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Substitua o *`my-cluster`* pelo nome do cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster. Substitua os valores de *`example value`* restantes pelos seus próprios valores. Os

nós são criados com a mesma versão do Kubernetes que o ambiente de gerenciamento, por padrão.

Antes de escolher um valor para `--node-type`, revise Escolher um tipo de instância de nó do Amazon EC2 ideal.

Substituir *my-key* pelo nome do par de chaves do Amazon EC2 ou da chave pública. Essa chave é usada para SSH em seus nós depois que eles forem iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no AWS Management Console. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2.

Crie seu grupo de nós com o comando a seguir.

> **Important**
>
> Caso queria implantar um grupo de nós no AWS Outposts, no Wavelength ou nas sub-redes de zonas locais, há outras considerações:
>
> - As sub-redes não devem ter passado quando você criou o cluster.
> - É necessário criar o grupo de nós com um arquivo config que especifique as sub-redes e `volumeType: gp2`. Para obter mais informações, consulte Criar um grupo de nós em um arquivo de configuração e Esquema de arquivo Config na documentação do `eksctl`.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --name al-nodes \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --ssh-access \
  --managed=false \
  --ssh-public-key my-key
```

Para implantar um grupo de nós que:

- Possa atribuir um número significativamente maior de endereços IP a Pods do que a configuração padrão, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2.
- Possa atribuir endereços `IPv4` a Pods de um bloco CIDR diferente do bloco da instância, consulte Rede personalizada para pods.
- pode atribuir endereços `IPv6` a Pods e serviços, consulte Endereços `IPv6` para clusters, Pods e services.
- Use o runtime `containerd`; é necessário implantar o grupo de nós usando um arquivo `config`. Para obter mais informações, consulte Teste a migração do Amazon Linux 2 de Docker para `containerd`.
- Não tenha acesso de saída à Internet, consulte Implementar clusters privados com acesso limitado à internet.

Para obter uma lista completa de todas as opções e padrões disponíveis, digite o comando a seguir.

```
eksctl create nodegroup --help
```

Se os nós não conseguirem se juntar ao cluster, consulte Worker nodes fail to join cluster (Falha nos nós ao ingressar no cluster) no manual de solução de problemas.

Veja um exemplo de saída abaixo. Várias linhas são emitidas enquanto os nós são criados. Uma das últimas linha de saída é o exemplo de linha a seguir.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opcional) Implante uma aplicação de amostra para testar o cluster e os nós do Linux.
4. Convém bloquear o acesso para Pod ao IMDS se as condições a seguir forem verdadeiras:

   - Você planeja atribuir perfis do IAM a todas as contas de serviço do Kubernetes para que os Pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum dos Pods do cluster requer acesso ao serviço de metadados de instâncias do Amazon EC2 (IMDS) por outros motivos, como recuperar a Região da AWS atual.

Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

AWS Management Console

Etapa 1: Para iniciar nós autogerenciados do Linux usando o AWS Management Console

1. Faça download da versão mais recente do modelo AWS CloudFormation.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

2. Aguarde até que o status do cluster seja exibido como `ACTIVE`. Se você executar os nós antes que o cluster esteja ativo, o registro dos nós no cluster falhará, e você terá de executá-los novamente.
3. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.
4. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos, padrão).
5. Para Specify template (Especificar modelo), selecione Upload a template file (Fazer upload de um arquivo de modelo) e depois Choose file (Escolher arquivo).
6. Selecione o arquivo `amazon-eks-nodegroup.yaml` que você baixou.
7. Escolha Próximo.
8. Na página Specify stack details (Especificar detalhes da pilha), insira os parâmetros de acordo e escolha Next (Próximo):
   - Stack name (Nome da pilha): escolha um nome de pilha para a pilha do AWS CloudFormation. Por exemplo, você pode chamar de ***`my-cluster-nodes`***. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
   - ClusterName: insira o nome que você usou ao criar o cluster do Amazon EKS. Esse nome deve ser igual ao nome do cluster, ou seus nós não poderão ingressar no cluster.

- ClusterControlPlaneSecurityGroup: escolha o valor de SecurityGroups no resultado do AWS CloudFormation que você gerou na criação da VPC.

  As etapas a seguir mostram uma operação para recuperar o grupo aplicável.

  1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
  2. Escolha o nome do cluster.
  3. Escolha a guia Redes.
  4. Use o valor de Additional security group (Grupos de segurança adicionais) como referência ao selecionar na lista suspensa ClusterControlPlaneSecurityGroup.
- NodeGroupName: insira um nome para o grupo de nós. Esse nome poderá ser usado superiormente para identificar o grupo de nós do Auto Scaling que for criado para os nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.
- NodeAutoScalingGroupMinSize: digite o número mínimo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeAutoScalingGroupDesiredCapacity: insira o número desejado de nós para o qual dimensionar quando a pilha for criada.
- NodeAutoScalingGroupMaxSize: digite o número máximo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeInstanceType: escolha um tipo de instância para os nós. Para obter mais informações, consulte Escolher um tipo de instância de nó do Amazon EC2 ideal.
- NodeImageIdSSMParam: pré-preenchido com o parâmetro do Amazon EC2 Systems Manager de uma AMI recente otimizada para Amazon EKS, para uma versão do Kubernetes variável. Para usar uma versão secundária diferente do Kubernetes compatível com o Amazon EKS, substitua `1.XX` por uma versão diferente compatível. Recomendamos especificar a mesma versão do Kubernetes que seu cluster.

  Você também pode substituir `amazon-linux-2` por um tipo de AMI diferente. Para obter mais informações, consulte Recuperar IDs de AMI do Amazon Linux recomendadas.

  > **Note**
  >
  > A AMI do nó do Amazon EKS é baseada no Amazon Linux. Você pode monitorar eventos de segurança ou privacidade para o Amazon Linux 2 no Centro de

segurança do Amazon Linux ou fazendo a assinatura do feed RSS associado. Os eventos de segurança e privacidade incluem uma visão geral do problema, quais pacotes são afetadas e como atualizar suas instâncias para corrigir o problema.

- NodeImageId: (opcional) se estiver usando sua própria AMI personalizada (em vez da AMI otimizada para o Amazon EKS), insira um ID de AMI do nó para a sua Região da AWS. Se você especificar um valor aqui, ele substituirá todos os valores no campo NodeImageIdSSMParam.
- NodeVolumeSize: especifique um tamanho de volume raiz para os nós, em GiB.
- NodeVolumeType: especifique um tipo de volume raiz para os nós.
- KeyName: insira o nome de um par de chaves SSH do Amazon EC2; que você pode usar para conectar-se usando SSH em seus nós, depois de iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no AWS Management Console. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2.

> **Note**
>
> Se você não fornecer um par de chaves aqui, ocorrerá uma falha ao criar a pilha do AWS CloudFormation.

- BootstrapArguments: especifique argumentos opcionais para passar para o script de bootstrap do nó, como argumentos `kubelet` adicionais. Para obter mais informações, consulte as bootstrap script usage information (informações sobre o uso do script de boostrap) no GitHub.

  Para implantar um grupo de nós que:

  - Possa atribuir um número significativamente maior de endereços IP a Pods do que a configuração padrão, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2.
  - Possa atribuir endereços `IPv4` a Pods de um bloco CIDR diferente do bloco da instância, consulte Rede personalizada para pods.
  - pode atribuir endereços `IPv6` a Pods e serviços, consulte Endereços `IPv6` para clusters, Pods e services.

  - Use o runtime `containerd`; é necessário implantar o grupo de nós usando um arquivo `config`. Para obter mais informações, consulte Teste a migração do Amazon Linux 2 de Docker para `containerd`.
  - Não tenha acesso de saída à Internet, consulte Implementar clusters privados com acesso limitado à internet.
- DisableIMDSv1: por padrão, cada nó oferece suporte ao serviço de metadados de instância versão 1 (IMDSv1) e IMDSv2. Você pode desabilitar IMDSv1. Para evitar que futuros nós e Pods no grupo de nós usem MDSv1, defina DisableIMDSv1 como true. Para obter mais informações, consulte Configuring the instance metadata service (Configurar o serviço de metadados de instância). Para obter mais informações sobre como restringir o acesso a ele em seus nós, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.
- VpcId: insira o ID da VPC que você criou.
- Sub-redes: escolha as sub-redes criadas para a sua VPC. Se você criou a VPC usando as etapas descritas em Criar uma VPC para o cluster do Amazon EKS, especifique apenas as sub-redes privadas na VPC para que os nós sejam executados. É possível ver quais sub-redes são privadas abrindo cada link de sub-rede na guia Networking (Redes) do cluster.

  > **Important**
  >
  > - Se alguma das sub-redes for pública, ela deverá ter a atribuição automática de atribuição de endereço IP público habilitada. Se a configuração não estiver habilitada para a sub-rede pública, os nós implantados nessa sub-rede pública não receberão um endereço IP público e não poderão se comunicar com o cluster nem com outros produtos da AWS. Se a sub-rede tiver sido implantada antes de 26 de março de 2020 usando qualquer um dos modelos doAWS CloudFormation VPC para o Amazon EKS, ou usando o `eksctl`, a atribuição automática de endereço IP público será desabilitada para sub-redes públicas. Para obter informações sobre como habilitar a atribuição de endereço IP público para uma sub-rede, consulte Modificar o atributo de endereçamento `IPv4` público para a sub-rede. Se o nó for implantado em uma sub-rede privada, ele poderá se comunicar com o cluster e com outros produtos da AWS por um gateway NAT.
  > - Se as sub-redes não tiverem acesso à Internet, certifique-se de que você esteja ciente das considerações e das etapas adicionais em Implementar clusters privados com acesso limitado à internet.

- Se você selecionou sub-redes do AWS Outposts, do Wavelength de Zonas locais, essas sub-redes não devem ter sido passadas quando você criou o cluster.

9. Selecione as opções desejadas na página Configure stack options (Configurar opções de pilha) e depois escolha Next (Próximo).
10. Marque a caixa de seleção à esquerda de I acknowledge that AWS CloudFormation might create IAM resources. (Reconheço que o pode criar recursos do IAM) e escolha Create stack (Criar pilha).
11. Quando a criação da pilha for concluída, selecione-a no console e escolha Outputs (Saídas).
12. Registre o valor de NodeInstanceRole para o grupo de nós criado. Você precisará dele ao configurar os nós do Amazon EKS.

Etapa 2: Habilitar os nós a ingressar no cluster

> **Note**
>
> Se você iniciou os nós dentro de uma VPC privada sem acesso de saída à Internet, certifique-se de habilitá-los para ingressar no cluster pela VPC.

1. Verifique se você já tem um `aws-auth` `ConfigMap`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se você receber um `aws-auth` `ConfigMap`, atualize-o conforme necessário.

   a. Abra o `ConfigMap` para edição.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Adicione uma nova `mapRoles` entrada conforme necessário. Defina o valor `rolearn` para o valor NodeInstanceRole que você registrou no procedimento anterior.

```
[...]
data:
```

```
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

c. Salve o arquivo e saia do seu editor de texto.

3. Se você recebeu um erro informando "`Error from server (NotFound): configmaps "aws-auth" not found`, aplique o estoque`ConfigMap`.

   a. Faça download do mapa de configuração.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-auth-cm.yaml
   ```

   b. No arquivo `aws-auth-cm.yaml`, configure o valor `rolearn` para o valor NodeInstanceRole que você registrou no procedimento anterior. É possível fazer isso com um editor de texto ou substituindo *`my-node-instance-role`* e executando o seguinte comando:

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
   ```

   c. Aplique a configuração. Esse comando pode demorar alguns minutos para ser concluído.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

4. Observe o status de seus nós e aguarde até que eles atinjam o status `Ready`.

   ```
   kubectl get nodes --watch
   ```

   Insira `Ctrl+C` para retornar a um prompt de shell.

> **Note**
>
> Se você receber qualquer erro de autorização ou de tipo de recurso, consulte Acesso negado ou não autorizado (`kubectl`) no tópico de solução de problemas.

Se os nós não conseguirem se juntar ao cluster, consulte Worker nodes fail to join cluster (Falha nos nós ao ingressar no cluster) no manual de solução de problemas.

5. (Somente nós de GPU) Se tiver escolhido um tipo de instância de GPU e a AMI acelerada otimizada para o Amazon EKS, você deverá aplicar o Plug-in de dispositivo NVIDIA para Kubernetes como um DaemonSet no cluster. Substitua *vX.X.X* pela versão desejada de NVIDIA/k8s-device-plugin antes de executar o comando a seguir.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/deployments/static/nvidia-device-plugin.yml
```

Etapa 3: Ações adicionais

1. (Opcional) Implante uma aplicação de amostra para testar o cluster e os nós do Linux.
2. (Opcional) Se a política do IAM gerenciada AmazonEKS_CNI_Policy (se você tiver um cluster IPv4) ou a *AmazonEKS_CNI_IPv6_Policy* (que você mesmo criou caso tenha um cluster IPv6) estiver anexada à the section called “Perfil do IAM do nó”, é recomendável atribuí-la a um perfil do IAM que você associará à conta de serviço aws-node do Kubernetes. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).
3. Convém bloquear o acesso para Pod ao IMDS se as condições a seguir forem verdadeiras:

   - Você planeja atribuir perfis do IAM a todas as contas de serviço do Kubernetes para que os Pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum dos Pods do cluster requer acesso ao serviço de metadados de instâncias do Amazon EC2 (IMDS) por outros motivos, como recuperar a Região da AWS atual.

   Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Criar nós autogerenciados com blocos de capacidade para ML

Os blocos de capacidade para machine learning (ML) permitem que você reserve instâncias com GPU para uma data futura, a fim de dar suporte a workloads de ML de curta duração. Para obter

mais informações, consulte Blocos de capacidade para ML no Guia do Usuário do Amazon EC2 para Instâncias Linux.

Considerações

> **Important**
>
> - Os blocos de capacidade só estão disponíveis para determinados tipos de instância do Amazon EC2 e Regiões da AWS. Para obter informações sobre compatibilidade, consulte Trabalhar com pré-requisitos de blocos de capacidade no Guia do usuário do Amazon EC2 para instâncias Linux.
> - No momento, os blocos de capacidade não podem ser usados com o Karpenter.
> - Se você criar o grupo de nós autogerenciados antes de a reserva de capacidade se tornar ativa, defina a capacidade desejada como `0`.
> - Para permitir tempo suficiente para a drenagem adequada do(s) nó(s), sugerimos que você agende o dimensionamento para reduzir para zero mais de 30 minutos antes do horário de término da reserva do bloco de capacidade.
> - Para que os Pods sejam drenados facilmente, recomendamos definir o AWS Node Termination Handler conforme explicado nas próximas etapas.

Usar blocos de capacidade com nós autogerenciados

Você pode usar blocos de capacidade com o Amazon EKS para provisionar e escalar seus nós autogerenciados. As etapas a seguir fornecem um exemplo geral de visão geral. Os exemplos de modelos do AWS CloudFormation não abrangem todos os aspectos necessários em uma workload de produção. Normalmente, você também vai querer um script de bootstrapping para unir o nó ao cluster, especificar a AMI acelerada do Amazon EKS e um perfil de instância apropriado para unir ao cluster. Para obter mais informações, consulte Criar nós autogerenciados do Amazon Linux.

1. Crie um modelo de execução que seja aplicável à workload. Para obter mais informações, consulte Usar blocos de capacidade para workloads de machine learning no Guia do usuário do Amazon EC2 Auto Scaling.

   Certifique-se de que `LaunchTemplateData` inclua o seguinte:

   - `InstanceMarketOptions` com `MarketType` definido como `"capacity-block"`

- `CapacityReservationSpecification: CapacityReservationTarget` com `CapacityReservationId` definido como bloco de capacidade (por exemplo:*cr-02168da1478b509e0*)
- `IamInstanceProfile` com o `Arn` definido como o *iam-instance-profile-arn* aplicável
- `ImageId` definido como o *image-id* aplicável
- `InstanceType` definido como um tipo de instância compatível com blocos de capacidade (por exemplo: *p5.48xlarge*)
- `SecurityGroupIds` definido como os IDs aplicáveis (por exemplo: *sg-05b1d815d1EXAMPLE*)
- `UserData` definido como o *user-data* aplicável para seu grupo de nós autogerenciados

Veja a seguir um trecho de um modelo do CloudFormation para criar um modelo de execução direcionado a um bloco de capacidade.

```
NodeLaunchTemplate:
  Type: "AWS::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      IamInstanceProfile:
        Arn: iam-instance-profile-arn
      ImageId: image-id
      InstanceType: p5.48xlarge
      KeyName: key-name
      SecurityGroupIds:
      - sg-05b1d815d1EXAMPLE
      UserData: user-data
```

Você deve passar a sub-rede na zona de disponibilidade na qual a reserva é feita porque os blocos de capacidade são zonais.

2. Use o modelo de execução para criar um grupo de nós autogerenciados. Se você estiver criando o grupo de nós autogerenciados antes de a reserva de capacidade se tornar ativa,

defina a capacidade desejada como `0`. Ao criar o grupo de nós, certifique-se de especificar apenas a sub-rede correspondente à Zona de Disponibilidade na qual a capacidade está reservada.

Veja abaixo um modelo do CloudFormation de exemplo que pode ser utilizado como referência ao criar um modelo que seja aplicável à sua workload. Esse exemplo obtém o `LaunchTemplateId` e a `Version` do recurso `AWS::Amazon EC2::LaunchTemplate` mostrado no exemplo anterior. Ele também obtém os valores para`DesiredCapacity`, `MaxSize`, `MinSize` e `VPCZoneIdentifier` que são declarados em outro lugar no mesmo modelo.

```
NodeGroup:
  Type: "AWS::AutoScaling::AutoScalingGroup"
  Properties:
    DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
    LaunchTemplate:
      LaunchTemplateId: !Ref NodeLaunchTemplate
      Version: !GetAtt NodeLaunchTemplate.LatestVersionNumber
    MaxSize: !Ref NodeAutoScalingGroupMaxSize
    MinSize: !Ref NodeAutoScalingGroupMinSize
    VPCZoneIdentifier: !Ref Subnets
    Tags:
      - Key: Name
        PropagateAtLaunch: true
        Value: !Sub ${ClusterName}-${NodeGroupName}-Node
      - Key: !Sub kubernetes.io/cluster/${ClusterName}
        PropagateAtLaunch: true
        Value: owned
```

3. Depois que o grupo de nós for criado com sucesso, certifique-se de registrar o `NodeInstanceRole` para o grupo de nós que foi criado. Você precisa disso para garantir que, quando o grupo de nós for escalado, os novos nós se juntem ao cluster e Kubernetes sejam capazes de reconhecer os nós. Para obter mais informações, consulte as instruções AWS Management Console em Criar nós autogerenciados do Amazon Linux.
4. Recomendamos que você crie uma política de escalabilidade programada para o grupo do Auto Scaling que se alinhe aos horários de reserva do bloco de capacidade. Para obter mais informações, consulte Escalabilidade agendada para o Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.

Você pode usar todas as instâncias reservadas até 30 minutos antes do horário final do bloco de capacidade. As instâncias que ainda estiverem em execução nesse momento começarão a ser encerradas. Para permitir tempo suficiente para a drenagem adequada do(s) nó(s), sugerimos que você agende o dimensionamento para reduzir para zero mais de 30 minutos antes do horário de término da reserva do bloco de capacidade.

Se, em vez disso, quiser aumentar a escala manualmente sempre que a reserva de capacidade chegar a `Active`, você precisará atualizar a capacidade desejada do grupo do Auto Scaling no horário de início da reserva do bloco de capacidade. Em seguida, você também precisaria reduzir a escala manualmente mais de 30 minutos antes do horário de término da reserva do bloco de capacidade.

5. O grupo de nós agora está pronto para workloads e Pods para ser programado.
6. Para que você Pods seja drenado normalmente, recomendamos que você configure o Manipulador do término do nó AWS. Esse manipulador poderá observar os eventos do ciclo de vida “ASG Scale-in” do Amazon EC2 Auto Scaling usando o EventBridge Kubernetes e permitir que o ajuste de escala automático execute as ações necessárias antes que a instância fique indisponível. Caso contrário, seus objetos Pods e Kubernetes ficarão presos em um estado pendente. Para obter mais informações, consulte Manipulador do término do nó da AWS no GitHub.

   Se você não configurar um Manipulador de término do nó, recomendamos que comece a drenar seus Pods manualmente antes de chegar à janela de 30 minutos, para que haja tempo suficiente para a drenagem adequada.

Criar nós Bottlerocket autogerenciados

> **Note**
>
> Os grupos de nós gerenciados podem oferecer algumas vantagens para seu caso de uso. Para obter mais informações, consulte Simplificar o ciclo de vida dos nós com grupos de nós gerenciados.

Este tópico descreve como iniciar grupos do Auto Scaling de nós do Bottlerocket que são registrados no cluster do Amazon EKS. O Bottlerocket é um sistema operacional de código aberto baseado no Linux da AWS que você pode utilizar para executar contêineres em máquinas virtuais ou hosts

bare metal. Depois que os nós ingressam no cluster, você pode implantar aplicações do Kubernetes neles. Para obter mais informações sobre o Bottlerocket, consulte Uso de uma AMI do Bottlerocket com o Amazon EKS no GitHub e Suporte para AMI personalizada na documentação do `eksctl`.

Para obter informações sobre as atualizações vigentes, consulte Operador de atualizações do Bottlerocket no GitHub.

Important

- Os nós do Amazon EKS são instâncias padrão do Amazon EC2, e você é cobrado por eles com base nos preços normais das instâncias do Amazon EC2. Para obter mais informações, consulte Definição de preço do Amazon EC2.
- Você pode iniciar nós do Bottlerocket em clusters estendidos do Amazon EKS em Outposts da AWS, mas não pode iniciá-los em clusters locais no AWS Outposts. Para obter mais informações, consulte Implantar o Amazon EKS on-premises com o AWS Outposts.
- É possível implantar em instâncias do Amazon EC2 com processadores `x86` ou Arm. Entretanto, não é possível implantar em instâncias que tenham chips Inferentia.
- Bottlerocket é compatível com AWS CloudFormation. No entanto, não há um modelo oficial do CloudFormation que possa ser copiado para implantar nós Bottlerocket para Amazon EKS.
- As imagens do Bottlerocket não vêm com um servidor ou shell SSH. Você pode usar métodos de acesso fora de banda para permitir que o SSH habilite o contêiner do administrador e especificar algumas etapas de configuração de bootstrapping com dados de usuário. Para obter mais informações, consulte essas seções no bottlerocket README.md no GitHub:
  - Exploration (Exploração)
  - Admin container (Contêiner Admin)
  - Configurações do Kubernetes

Para iniciar nós do Bottlerocket usando **`eksctl`**

Este procedimento exige a versão `eksctl` `0.187.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

> **Note**
>
> Este procedimento funciona apenas para clusters que foram criados com o `eksctl`.

1. Copie o conteúdo a seguir para o seu dispositivo. Substitua o *`my-cluster`* pelo nome do cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster. Substitua *`ng-bottlerocket`* por um nome para o seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Para implantar em instâncias do Arm, substitua *`m5.large`* por um tipo de instância Arm. Substitua *`my-ec2-keypair-name`* pelo nome de um par de chaves SSH do Amazon EC2; que você pode usar para conectar-se usando SSH em seus nós, depois de iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no AWS Management Console. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2. Substitua todos os *`example values`* restantes por seus próprios valores. Depois de fazer as substituições, execute o comando modificado para criar o arquivo `bottlerocket.yaml`.

   Se estiver especificando um tipo de instância do Amazon EC2 do Arm, analise as considerações em AMIs Amazon Linux Arm otimizadas para Amazon EKS antes de implantar. Para obter instruções sobre como implantar usando uma AMI personalizada, consulte Criação do Bottlerocket no GitHub e Suporte a AMI personalizada na documentação do `eksctl`. Para implantar um grupo de nós gerenciados, implante uma AMI personalizada usando um modelo de execução. Para obter mais informações, consulte Personalizar nós gerenciados com modelos de execução.

   > **Important**
   >
   > Para implantar um grupo de nós no AWS Outposts, no AWS Wavelength ou em sub-redes das Zonas locais da AWS, não passe o AWS Outposts, o AWS Wavelength ou

as sub-redes da Zona local da AWS ao criar o cluster. É necessário especificar as sub-redes no exemplo a seguir. Para obter mais informações, consulte Create a nodegroup from a config file (Criar um grupo de nós em um arquivo de configuração) e Config file schema (Esquema de arquivo de configuração) na documentação do `eksctl`. Substitua *`region-code`* pela Região da AWS em que está o cluster.

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.30'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-bottlerocket
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Bottlerocket
    ami: auto-ssm
    iam:
       attachPolicyARNs:
          - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

2. Implante os nós com o seguinte comando.

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

Veja um exemplo de saída abaixo.

Várias linhas são emitidas enquanto os nós são criados. Uma das últimas linha de saída é o exemplo de linha a seguir.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opcional) Crie um volume persistente do Kubernetes em um nó Bottlerocket usando o Plugin de CSI do Amazon EBS. O driver padrão do Amazon EBS depende de ferramentas do sistema de arquivos que não estão incluídas no Bottlerocket. Para obter mais informações sobre como criar uma classe de armazenamento usando o driver, consulte Armazene volumes do Kubernetes com o Amazon EBS.
4. (Opcional) Por padrão, o `kube-proxy` define o parâmetro do kernel `nf_conntrack_max` com um valor padrão que pode diferir do que o Bottlerocket originalmente define na inicialização. Para manter a configuração padrão do Bottlerocket, edite a configuração `kube-proxy` com o comando a seguir.

```
kubectl edit -n kube-system daemonset kube-proxy
```

Adicione `--conntrack-max-per-core` e `--conntrack-min` aos argumentos `kube-proxy` que estão no exemplo a seguir. Uma configuração de `0` não implica em nenhuma mudança.

```
      containers:
      - command:
        - kube-proxy
        - --v=2
        - --config=/var/lib/kube-proxy-config/config
        - --conntrack-max-per-core=0
        - --conntrack-min=0
```

5. (Opcional) Implante uma aplicação de amostra para testar os nós do Bottlerocket.
6. Convém bloquear o acesso para Pod ao IMDS se as condições a seguir forem verdadeiras:

   - Você planeja atribuir perfis do IAM a todas as contas de serviço do Kubernetes para que os Pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum dos Pods do cluster requer acesso ao serviço de metadados de instâncias do Amazon EC2 (IMDS) por outros motivos, como recuperar a Região da AWS atual.

Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Criar nós Microsoft Windows autogerenciados

Este tópico descreve como iniciar grupos do Auto Scaling de nós do Windows que são registrados no cluster do Amazon EKS. Depois que os nós ingressam no cluster, você pode implantar aplicações do Kubernetes neles.

Important

- Os nós do Amazon EKS são instâncias padrão do Amazon EC2, e você é cobrado por eles com base nos preços normais das instâncias do Amazon EC2. Para obter mais informações, consulte Definição de preço do Amazon EC2.
- Você pode iniciar nós do Windows em clusters estendidos do Amazon EKS no AWS Outposts, mas não pode iniciá-los em clusters locais no AWS Outposts. Para obter mais informações, consulte Implantar o Amazon EKS on-premises com o AWS Outposts.

Habilite o suporte ao Windows no cluster. Recomendamos revisar considerações importantes antes de iniciar um grupo de nós do Windows. Para obter mais informações, consulte Habilitar o suporte do Windows.

Você pode iniciar os nós auto-gerenciados do Windows com o `eksctl` ou o AWS Management Console.

eksctl

Para iniciar nós autogerenciados do Windows usando **`eksctl`**

Este procedimento exige que você instale o `eksctl` e que a versão do `eksctl` seja pelo menos a `0.187.0`. Você pode verificar a versão com o comando a seguir.

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

> **Note**
>
> Este procedimento funciona apenas para clusters que foram criados com o `eksctl`.

1. (Opcional) Se a política do IAM gerenciada AmazonEKS_CNI_Policy (se você tiver um cluster `IPv4`) ou a *`AmazonEKS_CNI_IPv6_Policy`* (que você mesmo criou caso tenha um cluster `IPv6`) estiver anexada à the section called “Perfil do IAM do nó”, é recomendável atribuí-la a um perfil do IAM que você associará à conta de serviço `aws-node` do Kubernetes. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).
2. Esse procedimento pressupõe que você tem um cluster existente. Caso ainda não tenha um cluster do Amazon EKS e um grupo de nós do Amazon Linux ao qual adicionar um grupo de nós do Windows, recomendamos que você siga o guia Conceitos básicos do Amazon EKS: `eksctl`. O guia fornece uma demonstração completa para criar um cluster do Amazon EKS com nós do Amazon Linux.

   Crie seu grupo de nós com o comando a seguir. Substitua *`region-code`* pela Região da AWS em que está o cluster. Substitua *`my-cluster`* pelo nome do cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster. Substitua *`ng-windows`* por um nome para o seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Para obter o Kubernetes versão `1.24` ou posterior, você pode substituir *`2019`* por `2022` para usar o Windows Server 2022. Substitua o resto dos *`example values`* pelos seus próprios valores.

   > **Important**
   >
   > Para implantar um grupo de nós em sub-redes do AWS Outposts, do AWS Wavelength ou de zonas locais da AWS, não passe as sub-redes do AWS Outposts, do Wavelength o de zonais locais ao criar o cluster. Crie o grupo de nós com um arquivo de configuração, especificando o AWS Outposts, o Wavelength ou sub-redes de Zonas locais. Para obter mais informações, consulte Criar um grupo de nós em

um arquivo de configuração e Esquema de arquivo Config na documentação do `eksctl`.

```
eksctl create nodegroup \
    --region region-code \
    --cluster my-cluster \
    --name ng-windows \
    --node-type t2.large \
    --nodes 3 \
    --nodes-min 1 \
    --nodes-max 4 \
    --managed=false \
    --node-ami-family WindowsServer2019FullContainer
```

Note

- Se os nós não conseguirem juntar-se ao cluster, consulte Worker nodes fail to join cluster (Falha nos nós ao ingressar no cluster) no manual Troubleshooting (Solução de problemas).
- Para ver as opções disponíveis para os comandos `eksctl`, insira o comando a seguir.

  ```
  eksctl command -help
  ```

Veja um exemplo de saída abaixo. Várias linhas são emitidas enquanto os nós são criados. Uma das últimas linha de saída é o exemplo de linha a seguir.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opcional) Implante uma aplicação de amostra para testar o cluster e os nós do Windows.
4. Convém bloquear o acesso para Pod ao IMDS se as condições a seguir forem verdadeiras:

   - Você planeja atribuir perfis do IAM a todas as contas de serviço do Kubernetes para que os Pods tenham apenas as permissões mínimas de que precisam.

- Nenhum dos Pods do cluster requer acesso ao serviço de metadados de instâncias do Amazon EC2 (IMDS) por outros motivos, como recuperar a Região da AWS atual.

Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

AWS Management Console

Pré-requisitos

- Um cluster existente do Amazon EKS e um grupo de nós do Linux. Se você não tiver esses recursos, recomendamos que siga um de nossos guias Começar a usar o Amazon EKS para criá-los. Os guias descrevem como criar um cluster do Amazon EKS com nós do Linux.
- Uma VPC e um grupo de segurança existentes que atendem aos requisitos de um cluster do Amazon EKS. Para ter mais informações, consulte Requisitos e considerações sobre a VPC e a sub-rede do Amazon EKS e Considerações e requisitos sobre grupos de segurança do Amazon EKS. O guia Começar a usar o Amazon EKS cria uma VPC que atende aos requisitos. Se preferir, você também pode seguir Criar uma VPC para o cluster do Amazon EKS para criar uma nova manualmente.
- Um cluster existente do Amazon EKS que usa uma VPC e um grupo de segurança que atendam aos requisitos de um cluster do Amazon EKS. Para obter mais informações, consulte Criar um cluster do Amazon EKS.. Se você tiver sub-redes na Região da AWS onde tenha o AWS Outposts, o AWS Wavelength ou o AWSLocal Zones habilitado, essas sub-redes não poderão ter sido passadas na criação do cluster.

Etapa 1: Para iniciar nós autogerenciados do Windows usando o AWS Management Console

1. Aguarde até que o status do cluster seja exibido como `ACTIVE`. Se você executar os nós antes que o cluster esteja ativo, o registro dos nós no cluster falhará, e você terá de executá-los novamente.
2. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.
3. Selecione Criar pilha.
4. Em Specify template, selecione Amazon S3 template URL (URL do modelo do Amazon S3).
5. Copie a URL a seguir e cole-a no Amazon S3 URL (URL do Amazon S3).

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. Escolha Next (Avançar) duas vezes.
7. Na página Quick create stack (Criar pilha rapidamente), insira os parâmetros a seguir de forma adequada:

   - Stack name (Nome da pilha): escolha um nome de pilha para a pilha do AWS CloudFormation. Por exemplo, você pode chamar de ***my-cluster*-nodes**.
   - ClusterName: insira o nome que você usou ao criar o cluster do Amazon EKS.

     > **Important**
     >
     > Esse nome deve corresponder exatamente ao nome usado em Etapa 1: Criar o cluster do Amazon EKS. Caso contrário, seus nós não poderão ser incorporados ao cluster.

   - ClusterControlPlaneSecurityGroup: escolha o grupo de segurança no resultado do AWS CloudFormation que você gerou na criação da VPC.

     As etapas a seguir mostram um método de recuperar o grupo aplicável.

     1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
     2. Escolha o nome do cluster.
     3. Escolha a guia Redes.
     4. Use o valor de Additional security group (Grupos de segurança adicionais) como referência ao selecionar na lista suspensa ClusterControlPlaneSecurityGroup.

   - NodeGroupName: insira um nome para o grupo de nós. Esse nome poderá ser usado superiormente para identificar o grupo de nós do Auto Scaling que for criado para os nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.
   - NodeAutoScalingGroupMinSize: digite o número mínimo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
   - NodeAutoScalingGroupDesiredCapacity: insira o número desejado de nós para o qual dimensionar quando a pilha for criada.

- NodeAutoScalingGroupMaxSize: digite o número máximo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeInstanceType: escolha um tipo de instância para os nós. Para obter mais informações, consulte Escolher um tipo de instância de nó do Amazon EC2 ideal.

> **Note**
>
> Os tipos de instâncias compatíveis com a versão mais recente do Amazon VPC CNI plugin for Kubernetesestão listados em vpc_ip_resource_limit.go no GitHub. Talvez seja necessário atualizar a versão da CNI para usar os tipos de instâncias compatíveis mais recentes. Para obter mais informações, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

- NodeImageIdSSMParam: preenchido com o parâmetro do Amazon EC2 Systems Manager do ID da AMI do Windows Core otimizada para Amazon EKS recomendada atualmente. Para usar a versão completa do Windows, substitua `Core` por `Full`.
- NodeImageId: (opcional) se estiver usando sua própria AMI personalizada (em vez da AMI otimizada para o Amazon EKS), insira um ID de AMI do nó para a sua Região da AWS. Se você especificar um valor para este campo, ele substituirá todos os valores no campo NodeImageIdSSMParam.
- NodeVolumeSize: especifique um tamanho de volume raiz para os nós, em GiB.
- KeyName: insira o nome de um par de chaves SSH do Amazon EC2; que você pode usar para conectar-se usando SSH em seus nós, depois de iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no AWS Management Console. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2.

> **Note**
>
> Se você não fornecer um par de chaves aqui, ocorrerá uma falha ao criar a pilha do AWS CloudFormation.

- BootstrapArguments: especifique argumentos opcionais para passar para o script de bootstrap do nó, como argumentos `kubelet` adicionais usando `-KubeletExtraArgs`.
- DisableIMDSv1: por padrão, cada nó oferece suporte ao serviço de metadados de instância versão 1 (IMDSv1) e IMDSv2. Você pode desabilitar IMDSv1. Para evitar que

futuros nós e Pods no grupo de nós usem MDSv1, defina DisableIMDSv1 como true. Para obter mais informações, consulte Configuring the instance metadata service (Configurar o serviço de metadados de instância).

- VpcId: selecione o ID para a VPC que você criou em .
- NodeSecurityGroups: selecione o grupo de segurança que foi criado para o grupo de nós do Linux quando você criou a VPC. Se os nós do Linux tiverem mais de um grupo de segurança anexado a eles, especifique todos eles. Isso é importante, por exemplo, quando o grupo de nós do Linux foi criado com `eksctl`.
- Subnets (Sub-redes): escolha as sub-redes que você criou. Se você criou a VPC usando as etapas em Criar uma VPC para o cluster do Amazon EKS, especifique apenas as sub-redes privadas na VPC para que os nós sejam executados.

> **Important**
>
> - Se alguma das sub-redes for pública, ela deverá ter a atribuição automática de atribuição de endereço IP público habilitada. Se a configuração não estiver habilitada para a sub-rede pública, os nós implantados nessa sub-rede pública não receberão um endereço IP público e não poderão se comunicar com o cluster nem com outros produtos da AWS. Se a sub-rede tiver sido implantada antes de 26 de março de 2020 usando qualquer um dos modelos doAWS CloudFormation VPC para o Amazon EKS, ou usando o `eksctl`, a atribuição automática de endereço IP público será desabilitada para sub-redes públicas. Para obter informações sobre como habilitar a atribuição de endereço IP público para uma sub-rede, consulte Modificar o atributo de endereçamento `IPv4` público para a sub-rede. Se o nó for implantado em uma sub-rede privada, ele poderá se comunicar com o cluster e com outros produtos da AWS por um gateway NAT.
> - Se as sub-redes não tiverem acesso à Internet, esteja ciente das considerações e das etapas adicionais em Implementar clusters privados com acesso limitado à internet.
> - Se você selecionou sub-redes do AWS Outposts, do Wavelength de Zonas locais, essas sub-redes não devem ter sido passadas quando você criou o cluster.

8. Confirme que a pilha pode criar recursos do IAM e escolha Create stack (Criar pilha).
9. Quando a criação da pilha for concluída, selecione-a no console e escolha Outputs (Saídas).

10. Registre o valor de NodeInstanceRole para o grupo de nós criado. Você precisará dele ao configurar os nós do Windows do Amazon EKS.

Etapa 2: Habilitar os nós a ingressar no cluster

1. Verifique se você já tem um `aws-auth` `ConfigMap`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se você receber um `aws-auth` `ConfigMap`, atualize-o conforme necessário.

   a. Abra o `ConfigMap` para edição.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Adicione novas `mapRoles` entradas conforme necessário. Defina os valores `rolearn` para os valores NodeInstanceRole que você registrou no procedimento anterior.

```
[...]
data:
  mapRoles: |
- rolearn: <ARN of linux instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: <ARN of windows instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
[...]
```

   c. Salve o arquivo e saia do seu editor de texto.

3. Se você recebeu um erro informando "`Error from server (NotFound): configmaps "aws-auth" not found`, aplique o estoque`ConfigMap`.

   a. Faça download do mapa de configuração.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/aws-auth-cm-windows.yaml
```

b. No arquivo `aws-auth-cm-windows.yaml`, configure os valores `rolearn` para os valores NodeInstanceRole aplicáveis que você registrou nos procedimentos anteriores. É possível fazer isso com um editor de texto ou substituindo o *`example values`* e executando o seguinte comando:

```
sed -i.bak -e 's|<ARN of linux instance role (not instance profile)>|my-
node-linux-instance-role|' \
    -e 's|<ARN of windows instance role (not instance profile)>|my-node-
windows-instance-role|' aws-auth-cm-windows.yaml
```

> **Important**
>
> - Não modifique outras linhas do arquivo.
> - Não use o mesmo perfil do IAM para nós do Windows e Linux.

c. Aplique a configuração. Esse comando pode levar alguns minutos para ser concluído.

```
kubectl apply -f aws-auth-cm-windows.yaml
```

4. Observe o status de seus nós e aguarde até que eles atinjam o status `Ready`.

```
kubectl get nodes --watch
```

Insira `Ctrl+C` para retornar a um prompt de shell.

> **Note**
>
> Se você receber qualquer erro de autorização ou de tipo de recurso, consulte Acesso negado ou não autorizado (`kubectl`) no tópico de solução de problemas.

Se os nós não conseguirem se juntar ao cluster, consulte Worker nodes fail to join cluster (Falha nos nós ao ingressar no cluster) no manual de solução de problemas.

Etapa 3: Ações adicionais

1. (Opcional) Implante uma aplicação de amostra para testar o cluster e os nós do Windows.
2. (Opcional) Se a política do IAM gerenciada AmazonEKS_CNI_Policy (se você tiver um cluster IPv4) ou a *AmazonEKS_CNI_IPv6_Policy* (que você mesmo criou caso tenha um cluster IPv6) estiver anexada à the section called “Perfil do IAM do nó”, é recomendável atribuí-la a um perfil do IAM que você associará à conta de serviço aws-node do Kubernetes. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).
3. Convém bloquear o acesso para Pod ao IMDS se as condições a seguir forem verdadeiras:

   - Você planeja atribuir perfis do IAM a todas as contas de serviço do Kubernetes para que os Pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum dos Pods do cluster requer acesso ao serviço de metadados de instâncias do Amazon EC2 (IMDS) por outros motivos, como recuperar a Região da AWS atual.

   Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Criar nós Ubuntu Linux autogerenciados

> **Note**
>
> Os grupos de nós gerenciados podem oferecer algumas vantagens para seu caso de uso. Para obter mais informações, consulte Simplificar o ciclo de vida dos nós com grupos de nós gerenciados.

Este tópico descreve como iniciar grupos do Auto Scaling de nós do Ubuntu no Amazon Elastic Kubernetes Service (EKS) ou do Ubuntu Pro no Amazon Elastic Kubernetes Service (EKS) que estão registrados no seu cluster do Amazon EKS. O Ubuntu e o Ubuntu Pro para EKS são baseados no Ubuntu Minimal LTS oficial, incluem o kernel da AWS personalizado, que é desenvolvido em conjunto com a AWS, e foram construídos especificamente para o EKS. O Ubuntu Pro adiciona cobertura de segurança adicional ao aceitar períodos de suporte estendidos do EKS, kernel livepatch, conformidade com FIPS e a capacidade de executar contêineres Pro ilimitados.

Depois que os nós ingressam no cluster, você pode implantar aplicações em contêiner neles. Para obter mais informações, acesse a documentação do Ubuntu na AWS e Suporte para AMIs personalizadas na documentação do `eksctl`.

> **Important**
>
> - Os nós do Amazon EKS são instâncias padrão do Amazon EC2, e você é cobrado por eles com base nos preços normais das instâncias do Amazon EC2. Para obter mais informações, consulte Definição de preço do Amazon EC2.
> - Você pode iniciar nós do Ubuntu em clusters estendidos do Amazon EKS no AWS Outposts, mas não pode iniciá-los em clusters locais no AWS Outposts. Para obter mais informações, consulte Implantar o Amazon EKS on-premises com o AWS Outposts.
> - É possível implantar em instâncias do Amazon EC2 com processadores `x86` ou Arm. No entanto, as instâncias que têm chips do Inferentia talvez precisem instalar o SDK do Neuron primeiro.

Para iniciar nós do Ubuntu para EKS ou do Ubuntu Pro para EKS usando o **`eksctl`**

Este procedimento exige a versão `eksctl` `0.187.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

> **Note**
>
> Este procedimento funciona apenas para clusters que foram criados com o `eksctl`.

1. Copie o conteúdo a seguir para o seu dispositivo. Substitua o `my-cluster` pelo nome do cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfabético e não pode ter mais de 100 caracteres. Substitua `ng-ubuntu` por um nome para o seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens

e sublinhados para os demais caracteres. Para implantar em instâncias do Arm, substitua `m5.large` por um tipo de instância Arm. Substitua `my-ec2-keypair-name` pelo nome de um par de chaves SSH do Amazon EC2; que você pode usar para conectar-se usando SSH em seus nós, depois de iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no AWS Management Console. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2. Substitua todos os *example values* restantes por seus próprios valores. Depois de fazer as substituições, execute o comando modificado para criar o arquivo `ubuntu.yaml`.

> **Important**
>
> Para implantar um grupo de nós no AWS Outposts, no AWS Wavelength ou em sub-redes das Zonas locais da AWS, não passe o AWS Outposts, o AWS Wavelength ou as sub-redes da Zona local da AWS ao criar o cluster. É necessário especificar as sub-redes no exemplo a seguir. Para obter mais informações, consulte Create a nodegroup from a config file (Criar um grupo de nós em um arquivo de configuração) e Config file schema (Esquema de arquivo de configuração) na documentação do `eksctl`. Substitua *region-code* pela Região da AWS em que está o cluster.

```
cat >ubuntu.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.30'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-ubuntu
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Ubuntu22.04
    ami: auto-ssm
    iam:
```

```
      attachPolicyARNs:
         - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
         - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
         - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
         - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
   ssh:
       allow: true
       publicKeyName: my-ec2-keypair-name
EOF
```

Para criar um grupo de nós do Ubuntu Pro, basta alterar o valor `amiFamily` de `UbuntuPro2204`.

2. Implante os nós com o seguinte comando.

```
eksctl create nodegroup --config-file=ubuntu.yaml
```

Veja um exemplo de saída abaixo.

Várias linhas são emitidas enquanto os nós são criados. Uma das últimas linha de saída é o exemplo de linha a seguir.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opcional) Implante uma aplicação de amostra para testar os nós do Ubuntu.
4. Convém bloquear o acesso para Pod ao IMDS se as condições a seguir forem verdadeiras:

   - Você planeja atribuir perfis do IAM a todas as contas de serviço do Kubernetes para que os Pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum dos Pods do cluster requer acesso ao serviço de metadados de instâncias do Amazon EC2 (IMDS) por outros motivos, como recuperar a Região da AWS atual.

   Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Atualizar nós autogerenciados para seu cluster

Quando uma nova AMI otimizada para Amazon EKS for lançada, considere substituir os nós no grupo de nós autogerenciados pela nova AMI. Da mesma forma, se você tiver atualizado a versão

do Kubernetes para o cluster do Amazon EKS, atualize os nós para usá-los com a mesma versão do Kubernetes.

> **Important**
>
> Este tópico aborda as atualizações do nó para nós autogerenciados. Se você estiver usando Simplificar o ciclo de vida dos nós com grupos de nós gerenciados, consulte Atualizar um grupo de nós gerenciados para seu cluster.

Há duas maneiras básicas de atualizar grupos de nós autogerenciados em seus clusters para usar uma nova AMI:

Migrar aplicações para um novo grupo de nós

Crie um novo grupo de nós e migre os Pods para esse grupo. Migrar para um novo grupo de nós é mais normal do que simplesmente atualizar o ID da AMI em uma pilha do AWS CloudFormation existente. Isso ocorre porque o processo de migração realiza taints no antigo grupo de nós, como `NoSchedule`, e drena os nós depois que uma nova pilha está pronta para aceitar a workload do Pod existente.

Atualizar uma pilha de nós do AWS CloudFormation

Para usar a nova AMI, atualize a pilha do AWS CloudFormation para um grupo de nós existente. Esse método não é compatível com grupos de nós que foram criados com o `eksctl`.

Migrar aplicações para um novo grupo de nós

Este tópico descreve como criar um novo grupo de nós, migrar facilmente as aplicações existentes para o novo grupo e remover o grupo de nós antigo do cluster. Você pode migrar para um novo grupo de nós usando o `eksctl` ou o AWS Management Console.

eksctl

Para migrar as aplicações para um novo grupo de nós com o **`eksctl`**

Para obter mais informações sobre o uso do eksctl para migração, consulte Unmanaged nodegroups (Grupos de nós não gerenciados) na documentação do `eksctl`.

Este procedimento exige a versão `eksctl` `0.187.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

> **Note**
>
> Este procedimento funciona apenas para clusters e grupos de nós que foram criados com o `eksctl`.

1. Recupere o nome dos grupos de nós existentes substituindo *`my-cluster`* pelo nome do cluster.

   ```
   eksctl get nodegroups --cluster=my-cluster
   ```

   Veja um exemplo de saída abaixo.

   ```
   CLUSTER      NODEGROUP          CREATED               MIN SIZE      MAX SIZE
    DESIRED CAPACITY     INSTANCE TYPE     IMAGE ID
   default      standard-nodes   2019-05-01T22:26:58Z  1             4             3
                       t3.medium         ami-05a71d034119ffc12
   ```

2. Inicie um novo grupo de nós com `eksctl` usando o comando a seguir. No comando, substitua cada *`example value`* por seus próprios valores. O número da versão não pode ser superior à versão do Kubernetes de seu ambiente de gerenciamento. Além disso, não pode ultrapassar duas versões secundárias anteriores à versão do Kubernetes de seu ambiente de gerenciamento. Recomendamos usar a mesma versão do ambiente de gerenciamento.

   Convém bloquear o acesso para Pod ao IMDS se as condições a seguir forem verdadeiras:

   - Você planeja atribuir perfis do IAM a todas as contas de serviço do Kubernetes para que os Pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum dos Pods do cluster requer acesso ao serviço de metadados de instâncias do Amazon EC2 (IMDS) por outros motivos, como recuperar a Região da AWS atual.

Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Para bloquear o acesso do Pod ao IMDS, adicione a opção `--disable-pod-imds` ao comando a seguir.

> **Note**
>
> Para obter mais sinalizadores disponíveis e suas descrições, consulte https://eksctl.io/.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --version 1.30 \
  --name standard-nodes-new \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --managed=false
```

3. Quando o comando anterior for concluído, verifique se todos os nós atingiram o estado `Ready` com o seguinte comando:

```
kubectl get nodes
```

4. Exclua cada grupo de nós original usando o comando a seguir. No comando, substitua cada *example value* pelos nomes de grupos de nós e de cluster:

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes-old
```

AWS Management Console and AWS CLI

Para migrar as aplicações para um novo grupo de nós com o AWS Management Console e a AWS CLI

1. Inicie um novo grupo de nós seguindo as etapas que estão descritas em Criar nós autogerenciados do Amazon Linux.
2. Quando a criação da pilha for concluída, selecione-a no console e escolha Outputs (Saídas).
3. Registre o valor de NodeInstanceRole para o grupo de nós criado. Você precisará dele para adicionar os novos nós do Amazon EKS ao cluster.

> **Note**
>
> Se você anexou qualquer outra política do IAM à função do IAM do antigo grupo de nós, anexe essas mesmas políticas à função do IAM do novo grupo de nós para manter essa funcionalidade no novo grupo. Isso se aplica a você se você adicionou permissões para o Kubernetes Cluster Autoscaler, por exemplo.

4. Atualize os grupos de segurança para ambos os grupos de nós, para que eles possam se comunicar um com o outro. Para obter mais informações, consulte Considerações e requisitos sobre grupos de segurança do Amazon EKS.

   a. Registre os IDs do grupo de segurança de ambos os grupos de nós. Isso é mostrado como o valor NodeSecurityGroup nas saídas da pilha do AWS CloudFormation.

      É possível usar os seguintes comandos da AWS CLI para obter os IDs do grupo de segurança dos nomes da pilha. Nesses comandos, `oldNodes` é o nome da pilha do AWS CloudFormation para a pilha de nós mais antiga, e `newNodes` é o nome da pilha para a qual você está migrando. Substitua cada *`example value`* por seus próprios valores.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name
 $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
```

```
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name
 $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
```

b. Adicione regras de entrada a cada grupo de segurança do nó para que eles aceitem o tráfego uns dos outros.

   Os comandos da AWS CLI a seguir adicionam regras de entrada a cada grupo de segurança que permite todo o tráfego em todos os protocolos do outro grupo de segurança. Isso permite que os Pods de cada grupo de nós se comuniquem entre si enquanto você estiver migrando a workload para o novo grupo.

```
aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 authorize-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

5. Edite o configmap do `aws-auth` para mapear a nova função da instância do nó no RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Adicione uma nova entrada `mapRoles` para o novo grupo do nós. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile)
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

Substitua o trecho *ARN of instance role (not instance profile)* pelo valor NodeInstanceRole que você registrou em uma etapa anterior. Então, salve e feche o arquivo para aplicar o configmap atualizado.

6. Observe o status dos nós e aguarde até que os novos nós ingressem no cluster e atinjam o status `Ready`.

```
kubectl get nodes --watch
```

7. (Opcional) Se você estiver usando o Cluster Autoscaler do Kubernetes, reduza a implantação para zero (0) réplica para evitar ações de escalabilidade conflitantes.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. Use o comando a seguir para aplicar taint em cada um dos nós que deseja remover com `NoSchedule`. Isso garante que novos Pods não sejam agendados ou reagendados nos nós que você está substituindo. Para obter mais informações, consulte Taints e Tolerâncias na documentação do Kubernetes.

```
kubectl taint nodes node_name key=value:NoSchedule
```

Se você estiver atualizando os nós para uma nova versão do Kubernetes, será possível identificar e aplicar taints em todos os nós de determinada versão do Kubernetes (neste caso, `1.28`) com o trecho de código a seguir. O número da versão não pode ser superior à versão do Kubernetes do seu ambiente de gerenciamento. Também não pode ultrapassar duas versões secundárias anteriores à versão do Kubernetes do ambiente de gerenciamento. Recomendamos usar a mesma versão do ambiente de gerenciamento.

```
K8S_VERSION=1.28
nodes=$(kubectl get nodes -o jsonpath="{.items[?
(@.status.nodeInfo.kubeletVersion==\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Tainting $node"
    kubectl taint nodes $node key=value:NoSchedule
done
```

9. Determine o provedor DNS do cluster.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

Veja um exemplo de saída abaixo. O cluster está usando CoreDNS para a resolução de DNS, mas, em vez disso, o cluster pode retornar `kube-dns`:

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

10. Se a implantação atual estiver executando menos de duas réplicas, expanda-a para duas réplicas. Substitua *`coredns`* por **kubedns** se a saída do comando anterior retornou isso.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11. Drene cada um dos nós que você deseja remover do cluster com o seguinte comando:

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

Se você estiver atualizando os nós para uma nova versão do Kubernetes, identifique e drene todos os nós de determinada versão do Kubernetes (neste caso, *`1.28`*) com o trecho de código a seguir.

```
K8S_VERSION=1.28
nodes=$(kubectl get nodes -o jsonpath="{.items[?
(@.status.nodeInfo.kubeletVersion==\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-local-data
done
```

12. Depois que os antigos nós forem esvaziados, revogue as regras de entrada do grupo de segurança que você autorizou anteriormente. Então, exclua a pilha do AWS CloudFormation para terminar as instâncias.

> **Note**
>
> Se anexou qualquer política adicional do IAM ao perfil do IAM no antigo grupo de nós (como adicionar permissões para o Cluster Autoscaler do Kubernetes),

desvincule essas políticas adicionais da função antes de excluir a pilha do AWS CloudFormation.

a. Revogue as regras de entrada criadas anteriormente para os grupos de segurança do nó. Nesses comandos, `oldNodes` é o nome da pilha do AWS CloudFormation para a pilha de nós mais antiga, e `newNodes` é o nome da pilha para a qual você está migrando.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name
 $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name
 $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 revoke-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

b. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.

c. Selecione a pilha antiga do nó.

d. Escolha Excluir.

e. Na caixa de diálogo de confirmação Delete stack (Excluir pilha), escolha Delete stack (Excluir pilha).

13. Edite o configmap `aws-auth` para remover a antiga função da instância do nó do RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Exclua a entrada `mapRoles` do antigo grupo nós. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-
W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-15-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
```

Salve e feche o arquivo para aplicar o configmap atualizado.

14. (Opcional) Se você estiver usando o Kubernetes Cluster Autoscaler, reduza a implantação para uma réplica.

> **Note**
>
> Você também deve etiquetar o novo grupo do Auto Scaling corretamente (por exemplo, `k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/`*`my-cluster`*) e atualizar o comando da implantação do Cluster Autoscaler de modo a apontar para o grupo do Auto Scaling recém-marcado. Para obter mais informações, consulte Autoscaler do cluster na AWS.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15. (Opcional) Verifique se você está usando a versão mais recente do plugin CNI da Amazon VPC para Kubernetes. Talvez seja necessário atualizar a versão da CNI para usar os tipos

de instâncias compatíveis mais recentes. Para obter mais informações, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

16. Se o cluster estiver usando o `kube-dns` para resolução de DNS (consulte a etapa anterior), reduza a implantação de `kube-dns` para uma réplica.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

Atualizar uma pilha de nós do AWS CloudFormation

Este tópico descreve como atualizar uma pilha de nós autogerenciados do AWS CloudFormation com uma nova AMI. Você pode usar esse procedimento para atualizar os nós para uma nova versão do Kubernetes após uma atualização do cluster. Senão, é possível atualizar para a AMI otimizada do Amazon EKS mais recente para uma versão existente do Kubernetes.

> **Important**
>
> Este tópico aborda as atualizações do nó para nós autogerenciados. Para obter informações sobre como utilizar o Simplificar o ciclo de vida dos nós com grupos de nós gerenciados, consulte Atualizar um grupo de nós gerenciados para seu cluster.

O modelo do AWS CloudFormation mais recente do nó padrão do Amazon EKS é configurado para executar uma instância com a nova AMI no cluster, antes de remover um antigo, um de cada vez. Essa configuração garante que você esteja com a contagem desejada do grupo do Auto Scaling de instâncias ativas no cluster durante a implantação da atualização.

> **Note**
>
> Esse método não é compatível com grupos de nós que foram criados com o `eksctl`. Se você criou o cluster ou o grupo de nós com o `eksctl`, consulte Migrar aplicações para um novo grupo de nós.

Para atualizar um grupo de nós existente

1. Determine o provedor DNS de seu cluster.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

Veja um exemplo de saída abaixo. O cluster está usando CoreDNS para a resolução de DNS, mas, em vez disso, o cluster pode retornar `kube-dns`. Sua saída pode parecer diferente dependendo da versão do `kubectl` que você está usando.

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

2. Se a implantação atual estiver executando menos de duas réplicas, expanda-a para duas réplicas. Substitua *coredns* por **kube-dns** se a saída do comando anterior retornou isso.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

3. (Opcional) Se você estiver usando o Cluster Autoscaler do Kubernetes, reduza a implantação para zero (0) réplica para evitar ações de escalabilidade conflitantes.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

4. Determine o tipo de instância e a contagem de instâncias desejada do grupo de nós atual. Você poderá inserir esses valores superiormente ao atualizar o modelo do AWS CloudFormation para o grupo.

   a. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.

   b. No painel de navegação à esquerda, escolha Launch Configurations (Configurações de execução) e anote o tipo de instância da configuração de execução do nó existente.

   c. No painel de navegação à esquerda, escolha Auto Scaling Groups (Grupos do Auto Scaling) e observe a contagem de instâncias Desired (Desejadas) para o grupo do Auto Scaling do nó existente.

5. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.

6. Selecione a pilha do grupo de nós e escolha Update (Atualizar).

7. Selecione Replace current template (Substituir modelo atual) e selecione Amazon S3 URL (URL do Simple Storage Service (Amazon S3)).

8. Em Amazon S3 URL (URL do Amazon S3), cole o URL a seguir na área de texto para garantir que você esteja usando a versão mais recente do modelo do AWS CloudFormation do nó. Depois, escolha Next (Próximo):

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

9. Na página Specify stack details (Especificar detalhes da pilha), preencha os parâmetros a seguir e escolha Next (Próximo):

   - NodeAutoScalingGroupDesiredCapacity: insira a contagem de instâncias desejada que você registrou em uma etapa anterior. Ou insira o novo número desejado de nós para o qual escalar quando a pilha for criada.
   - NodeAutoScalingGroupMaxSize: digite o número máximo de nós para o qual o grupo Auto Scaling de nós pode ser aumentado. Esse valor deve ser pelo menos um nó a mais do que a capacidade desejada. Assim, você pode realizar uma atualização contínua dos nós sem reduzir a contagem de nós durante a atualização.
   - NodeInstanceType: escolha o tipo de instância registrada em uma etapa anterior. Se preferir, escolha um tipo de instância diferente para os nós. Antes de escolher um tipo de instância diferente, revise Escolher um tipo de instância de nó do Amazon EC2 ideal. Cada tipo de instância do Amazon EC2 suporta um número máximo de interfaces elásticas de rede (interfaces de rede) e cada interface de rede oferece suporte a um número máximo de endereços IP. Como cada nó de processamento e Pod recebe seu próprio endereço IP, é importante escolher um tipo de instância que ofereça suporte ao número máximo de Pods que você deseja executar em cada nó do Amazon EC2. Para obter uma lista do número de interfaces de rede e endereços IP compatíveis com tipos de instância, consulte Endereços IP por interface de rede por tipo de instância. Por exemplo, os tipos de instância `m5.large` oferecem suporte a no máximo 30 endereços IP para o nó de processamento e Pods.

     > **Note**
     >
     > Os tipos de instâncias compatíveis com a versão mais recente do Amazon VPC CNI plugin for Kubernetes estão listados em vpc_ip_resource_limit.go no GitHub. Talvez seja necessário atualizar a versão do Amazon VPC CNI plugin for Kubernetes para usar os tipos de instâncias compatíveis mais recentes. Para obter mais informações, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

> **Important**
>
> Alguns tipos de instância podem não estar disponíveis em cada Regiões da AWS.

- NodeImageIdSSMParam – O parâmetro do Amazon EC2 Systems Manager do ID da AMI para o qual você deseja atualizar. O valor a seguir usa a AMI otimizada para o Amazon EKS mais recente para a versão `1.30` do Kubernetes.

```
/aws/service/eks/optimized-ami/1.30/amazon-linux-2/recommended/image_id
```

  É possível substituir *`1.30`* por uma versão compatível do Kubernetes que seja a mesma. Ou deve ser até uma versão anterior à versão do Kubernetes em execução no ambiente de gerenciamento. Recomendamos que você mantenha seus nós na mesma versão do plano de controle. Você também pode substituir *`amazon-linux-2`* por um tipo de AMI diferente. Para obter mais informações, consulte Recuperar IDs de AMI do Amazon Linux recomendadas.

> **Note**
>
> O uso do parâmetro do Amazon EC2 Systems Manager permite atualizar os nós no futuro sem precisar pesquisar e especificar um ID de AMI. Se a pilha do AWS CloudFormation estiver usando esse valor, qualquer atualização da pilha sempre executará a AMI otimizada para o Amazon EKS recomendada para a versão do Kubernetes especificada. Isso se aplica mesmo que você não altere nenhum valor no template.

- NodeImageId – para usar sua própria AMI personalizada, insira o ID da AMI a ser usada.

> **Important**
>
> Esse valor substitui qualquer valor especificado para NodeImageIdSSMParam. Se você quiser usar o valor de NodeImageIdSSMParam, verifique se o valor de NodeImageId está em branco.

- DisableIMDSv1: por padrão, cada nó oferece suporte ao serviço de metadados de instância versão 1 (IMDSv1) e IMDSv2. No entanto, você pode desabilitar o IMDSv1. Selecione true se você não quiser que os nós ou Pods agendados no grupo de nós usem IMDSv1. Para obter

mais informações, consulte Configuring the instance metadata service (Configurar o serviço de metadados de instância). Se você implementou perfis do IAM para contas de serviço e atribuiu as permissões necessárias diretamente a todos os Pods que exigem acesso aos serviços da AWS. Desse modo, nenhum dos Pods do cluster requer acesso ao IMDS por outros motivos, como, por exemplo, para recuperar a Região da AWS atual. Em seguida, você também pode desabilitar o acesso ao IMDSv2 para Pods que não usam redes de host. Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

10. (Opcional) Na página Options (Opções), marque os recursos da pilha. Escolha Próximo.
11. Na página Review (Revisão), analise suas informações, confirme se a pilha pode criar recursos do IAM e selecione Update stack (Atualizar pilha).

> **Note**
>
> A atualização de cada nó no cluster leva vários minutos. Aguarde até que a atualização de todos os nós seja concluída antes de executar as próximas etapas.

12. Se o provedor de DNS do cluster for `kube-dns`, reduza a implantação `kube-dns` para uma réplica.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13. (Opcional) Se você estiver usando o Kubernetes Cluster Autoscaler, reduza a implantação para a quantidade de réplicas desejada.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

14. (Opcional) Verifique se você está usando a versão mais recente do Amazon VPC CNI plugin for Kubernetes. Talvez seja necessário atualizar a versão do Amazon VPC CNI plugin for Kubernetes para usar os tipos de instâncias compatíveis mais recentes. Para obter mais informações, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

Simplificar o gerenciamento da computação com o AWS Fargate

> **Important**
>
> O AWS Fargate com o Amazon EKS não está disponível para a AWS GovCloud (Leste dos EUA) e AWS GovCloud (Oeste dos EUA).

Este tópico discute o uso do Amazon EKS para executar Kubernetes do Pods no AWS Fargate. O Fargate é uma tecnologia que fornece capacidade computacional sob demanda do tamanho certo para contêineres. Com o Fargate, você não tem que provisionar, configurar ou escalar grupos de máquinas virtuais você mesmo para executar contêineres. Você também não precisa escolher tipos de servidor, decidir quando escalar grupos de nós ou otimizar o empacotamento dos clusters.

É possível controlar quais Pods serão iniciados no Fargate e como serão executados com os perfis do Fargate. Os perfis do Fargate são definidos como parte do seu cluster do Amazon EKS. O Amazon EKS integra o Kubernetes ao Fargate com controladores criados pela AWS usando o modelo extensível upstream fornecido pelo Kubernetes. Esses controladores são executados como parte do ambiente de gerenciamento do Kubernetes gerenciado pelo Amazon EKS e são responsáveis por agendar Pods nativos do Kubernetes. Os controladores do Fargate incluem um novo agendador que é executado com o agendador do Kubernetes, além de vários controladores de admissão de mutação e de validação. Quando você inicia um Pod que atende aos critérios de execução no Fargate, os controladores do Fargate em execução no cluster reconhecem, atualizam e agendar o Pod no Fargate.

Este tópico descreve os diferentes componentes dos Pods que são executados no Fargate e faz considerações especiais para o uso do Fargate com o Amazon EKS.

Considerações sobre o AWS Fargate

Veja a seguir algumas considerações sobre o uso do Fargate no Amazon EKS.

- Cada Pod executado no Fargate tem sua própria barreira de isolamento. Eles não compartilham o kernel subjacente, os recursos de CPU, os recursos de memória nem a interface de rede elástica com outro Pod.
- Os balanceadores de carga da rede e os balanceadores de carga da aplicação (ALBs) só podem ser usados com o Fargate com destinos IP. Para ter mais informações, consulte Criar um

balanceador de carga da rede e Aplicação de roteamento e tráfego HTTP com Application Load Balancers.

- Os serviços expostos ao Fargate são executados somente no modo IP do tipo de destino e não no modo IP do nó. A maneira recomendada de verificar a conectividade de um serviço em execução em um nó gerenciado e um serviço em execução no Fargate é se conectar pelo nome de serviço.
- Os pods devem corresponder a um perfil do Fargate no momento em que são agendados para execução no Fargate. Pods que não correspondam a um perfil do Fargate podem ficar retidos como `Pending`. Se houver um perfil do Fargate correspondente, será possível excluir os Podspendentes criados para serem regendados no Fargate.
- Não há suporte a Deamonsets no Fargate. Se a aplicação exigir um daemon, reconfigure esse daemon para ser executado como um contêiner associado nos Pods.
- Não há suporte a contêineres privilegiados no Fargate.
- Os pods que são executados no Fargate não podem especificar `HostPort` ou `HostNetwork` no manifesto do Pod.
- O limite flexível de `nofile` e `nproc` padrão é 1.024, e o limite rígido é 65.535 para Pods do Fargate.
- No momento, GPUs não estão disponíveis no Fargate.
- Só há suporte a pods em execução no Fargate em sub-redes privadas (com acesso de gateway NAT aos serviços da AWS, mas não uma rota direta para um gateway da Internet), portanto, a VPC do cluster deve ter sub-redes privadas disponíveis. Para clusters sem acesso de saída à Internet, consulte Implementar clusters privados com acesso limitado à internet.
- É possível usar Ajustar os recursos de pods com o Vertical Pod Autoscaler para definir o tamanho correto da CPU e da memória para os Pods do Fargate e, em seguida, usar Escalar as implantações de pods com o Horizontal Pod Autoscaler para dimensionar a escala desses Pods. Se quiser que o Vertical Pod Autoscaler reimplante automaticamente os Pods no Fargate com combinações maiores de CPU e memória, defina o modo do Vertical Pod Autoscaler como `Auto` ou `Recreate` para garantir a funcionalidade correta. Para obter mais informações, consulte a documentação do Vertical Pod Autoscaler no GitHub.
- A resolução DNS e os nomes de host DNS devem estar habilitados para sua VPC. Para obter mais informações, consulte Visualizar e atualizar o suporte DNS para VPC.
- O Fargate do Amazon EKS adiciona defesa completa para aplicações do Kubernetes isolando cada pod em uma máquina virtual (VM). Essa barreira de VM impede o acesso aos recursos baseados em host usados por outros pods, no caso de um escape de contêiner, um método comum de atacar aplicações em contêiner e obter acesso a recursos fora do contêiner.

O uso do Amazon EKS não altera suas responsabilidades no modelo de responsabilidade compartilhada. Você deve considerar cuidadosamente a configuração de controles de governança e segurança do cluster. A maneira mais segura de isolar uma aplicação é sempre executá-la em um cluster separado.

- Os perfis do Fargate são compatíveis com a especificação de sub-redes de blocos CIDR secundários da VPC. Talvez você queira especificar um bloco CIDR secundário. Isso porque existe um número limitado de endereços IP disponíveis em uma sub-rede. Como resultado, existe também um número limitado de Pods que podem ser criados no cluster. Usando diferentes sub-redes para os Pods, você pode aumentar o número de endereços IP disponíveis. Para obter mais informações, consulte Adicionar blocos CIDR `IPv4` a uma VPC.
- O serviço de metadados de instância do Amazon EC2 (IMDS) não está disponível para Pods implantados em nós do Fargate. Se você tiver Pods implantados no Fargate que precisem de credenciais do IAM, atribua-as aos Pods usando Perfis do IAM para contas de serviço. Se os Pods precisarem de acesso a outras informações disponíveis por meio do IMDS, você deve fazer a codificação rígida dessas informações na especificação do Pod. Isso inclui a Região da AWS ou a zona de disponibilidade em que um Pod é implantado.
- Não é possível implantar Pods do Fargate no AWS Outposts, AWS Wavelength ou AWS Local Zones.
- O Amazon EKS deve aplicar patches nos Pods periodicamente para mantê-los seguros. Tentamos fazer atualizações de uma maneira que reduza o impacto, mas há situações em que os Pods deverão ser excluídos se não forem removidos com êxito. Há algumas medidas que você pode tomar para minimizar a interrupção. Para obter mais informações, consulte Definir ações para eventos de correção do sistema operacional do AWS Fargate.
- O plugin CNI do Amazon VPC para Amazon EKS é instalado em nós do Fargate. Você não pode usar Plugins CNI compatíveis alternativos com nós do Fargate.
- Um Pod em execução no Fargate monta automaticamente um sistema de arquivos do Amazon EFS. Você não pode usar o provisionamento dinâmico de volume persistente com nós do Fargate, mas pode usar o provisionamento estático.
- Você não pode montar volumes do Amazon EBS em Pods do Fargate.
- Você pode executar o controlador da CSI do Amazon EBS em nós do Fargate, mas o DaemonSet do nó da CSI do Amazon EBS só pode ser executado em instâncias do Amazon EC2.

- Depois que um Kubernetes Job é marcado `Completed` ou `Failed`, o Pods que ele Job cria normalmente continua existindo. Esse comportamento permite que você visualize seus registros e resultados, mas com o Fargate você incorrerá em custos se não os limpar posteriormente Job.

  Para excluir automaticamente o relacionado Pods após uma Job conclusão ou falha, você pode especificar um período de tempo usando o controlador de tempo de vida (TTL). O exemplo a seguir mostra a especificação `.spec.ttlSecondsAfterFinished` em seu Job manifesto.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  template:
    spec:
      containers:
      - name: busybox
        image: busybox
        command: ["/bin/sh", "-c", "sleep 10"]
      restartPolicy: Never
  ttlSecondsAfterFinished: 60 # <-- TTL controller
```

Conceitos básicos do AWS Fargate para o seu cluster

> **Important**
>
> O AWS Fargate com o Amazon EKS não está disponível para a AWS GovCloud (Leste dos EUA) e AWS GovCloud (Oeste dos EUA).

Este tópico descreve como começar a executar os Pods no AWS Fargate com o cluster do Amazon EKS.

Se você restringir o acesso ao endpoint público do cluster usando blocos CIDR, é recomendável também habilitar o acesso ao endpoint privado. Dessa forma, os Pods do Fargate podem se comunicar com o cluster. Sem o endpoint privado ativado, os blocos CIDR especificados para acesso público devem incluir as origens de saída da VPC. Para obter mais informações, consulte Controlar o acesso à rede ao endpoint do servidor de API do cluster.

Pré-requisito

Um cluster existente. Se você ainda não tem um cluster do Amazon EKS, consulte Começar a usar o Amazon EKS.

Etapa 1: verificar se os nós existentes podem se comunicar com os Pods do Fargate

Se estiver trabalhando com um novo cluster sem nós ou com um cluster que tenha somente grupos de nós gerenciados, você pode avançar para Etapa 2: criar um perfil de execução de Pod do Fargate.

Pressuponha que você esteja trabalhando com um cluster existente que já tenha nós associados a ele. Verifique se os Pods nesses nós podem se comunicar livremente com os Pods que estão em execução no Fargate. Os Pods que estão em execução no Fargate são automaticamente configurados para usar o grupo de segurança do cluster ao qual estão associados. Garanta que qualquer nó existente no cluster possa enviar e receber tráfego entre o grupo de segurança do cluster. Como Simplificar o ciclo de vida dos nós com grupos de nós gerenciados são configurados automaticamente para usar o grupo de segurança do cluster, não é necessário modificá-los nem conferir sua compatibilidade.

Para grupos de nós existentes que foram criados com `eksctl` ou modelos de AWS CloudFormation gerenciados pelo Amazon EKS, você pode adicionar o grupo de segurança de cluster aos nós manualmente. Ou então, pode modificar o modelo de lançamento do grupo do Auto Scaling para o grupo de nós para anexar o grupo de segurança do cluster às instâncias. Para obter mais informações, consulte Alterar os grupos de segurança de uma instância no Guia do usuário da Amazon VPC.

Você pode conferir se há um grupo de segurança para o seu cluster no AWS Management Console, na seção Networking (Rede). Ou pode usar o comando a seguinte comando de AWS CLI. Quando usar esse comando, substitua *`my-cluster`* pelo nome de seu cluster.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Etapa 2: criar um perfil de execução de Pod do Fargate

Quando o cluster cria Pods no AWS Fargate, os componentes que são executados na infraestrutura do Fargate devem fazer chamadas para as APIs da AWS em seu nome. A função de execução de

Pod do Amazon EKS fornece as permissões do IAM para isso. Para criar uma função de execução de Pod de AWS Fargate, consulte Perfil do IAM para execução de Pod do Amazon EKS.

> **Note**
>
> Se você criou o cluster com o `eksctl` usando a opção `--fargate`, o cluster já tem uma função de execução de Pod que é possível encontrar no console do IAM com o padrão `eksctl-my-cluster-FargatePodExecutionRole-ABCDEFGHIJKL`. Da mesma forma, se você usar o `eksctl` para criar os perfis do Fargate, o `eksctl` criará a função de execução de Pod, caso ela ainda não tenha sido criada.

Etapa 3: criar um perfil do Fargate para o cluster

Antes de agendar os Pods em execução no Fargate no cluster, é necessário definir um perfil do Fargate que especifique quais Pods deverão usá-lo quando forem executados. Para obter mais informações, consulte Defina quais Pods usarão o AWS Fargate quando em execução.

> **Note**
>
> Se você criou o cluster com o `eksctl` usando a opção `--fargate`, um perfil do Fargate já foi criado para o cluster com seletores para todos os Pods nos namespaces `kube-system` e `default`. Use o procedimento a seguir para criar perfis do Fargate para qualquer outro namespace que você deseja usar com o Fargate.

Você pode criar um perfil do Fargate usando o `eksctl` ou o AWS Management Console.

eksctl

Este procedimento exige a versão `eksctl` `0.187.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Para criar um perfil do Fargate com o **`eksctl`**

Crie o perfil do Fargate com o comando `eksctl` a seguir, substituindo cada *example value* por seus próprios valores. Você precisa especificar um namespace. Mas, a opção `--labels` não é exigida.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

É possível usar certos curingas para rótulos *my-kubernetes-namespace* e *key=value* . Para obter mais informações, consulte Curingas do perfil do Fargate.

AWS Management Console

Como criar um perfil do Fargate para um cluster com o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha o cluster para o qual deve ser criado um perfil do Fargate.
3. Escolha a guia Compute (Computação).
4. Em Fargate profile (Perfil do Fargate), escolha Add Fargate profile (Adicionar perfil do Fargate).
5. Na página Configure Fargate profile (Configurar o perfil do Fargate), faça o seguinte:
   a. Em Name (Nome), insira um nome para o perfil do Fargate. O nome deve ser exclusivo.
   b. Em Pod execution role (Função de execução de pod), escolha a função de execução de Pod que deve ser usada com o perfil do Fargate. Somente funções do IAM com o principal de serviço do `eks-fargate-pods.amazonaws.com` são mostradas. Se nenhuma função estiver listada, você deve criar uma. Para obter mais informações, consulte Perfil do IAM para execução de Pod do Amazon EKS.
   c. Modifique as sub-redes selecionadas conforme necessário.

> **Note**
>
> Somente sub-redes privadas são compatíveis com Pods em execução no Fargate.

d. Em Tags (Etiquetas), você tem a opção de etiquetar o perfil do Fargate. Essas tags não são propagadas para outros recursos associados ao perfil, como Pods.

e. Escolha Próximo.

6. Na página Configure Pod selection (Configurar a seleção de), faça o seguinte:

a. Em Namespace, insira um namespace para fazer a correspondência com Pods.

- É possível usar namespaces específicos para a correspondência, como **`kube-system`** ou **`default`**.
- É possível usar certos curingas (por exemplo, **`prod-*`**) para fazer a correspondência com vários namespaces (por exemplo, `prod-deployment` e `prod-test`). Para obter mais informações, consulte Curingas do perfil do Fargate.

b. (Opcional) Adicione rótulos Kubernetes ao seletor. Adicione-os especificamente ao seletor ao qual os Pods no namespace especificado precisam corresponder.

- É possível adicionar o rótulo **`infrastructure: fargate`** ao seletor para que somente Pods no namespace especificado que também tenham o rótulo Kubernetes do `infrastructure: fargate` correspondam ao seletor.
- É possível usar certos curingas (por exemplo, **`key?: value?`**) para fazer a correspondência com vários namespaces (por exemplo, `keya: valuea` e `keyb: valueb`). Para obter mais informações, consulte Curingas do perfil do Fargate.

c. Escolha Próximo.

7. Na página Review and create (Revisar e criar), revise as informações do perfil do Fargate e selecione Create (Criar).

Etapa 4: atualizar o CoreDNS

Por padrão, o CoreDNS é configurado para ser executado na infraestrutura do Amazon EC2 nos clusters do Amazon EKS. Se você só quiser executar os Pods no Fargate no cluster, conclua as etapas a seguir.

> **Note**
>
> Se você criou o cluster com o `eksctl` usando a opção `--fargate`, poderá avançar para Próximas etapas.

1. Crie um perfil do Fargate para o CoreDNS com o comando a seguir. Substitua *my-cluster* pelo nome do cluster, *111122223333* pelo ID da conta, *AmazonEKSFargatePodExecutionRole* pelo nome da função de execução de Pod e *0000000000000001*, *0000000000000002* e *0000000000000003* pelos IDs das sub-redes privadas. Se você não tiver uma função de execução de Pod, será necessário criar uma primeiro.

   > **Important**
   >
   > O ARN da função não pode incluir um caminho diferente de `/`. Por exemplo, se o nome da sua função for `development/apps/my-role`, você precisará alterá-lo para `my-role` ao especificar o ARN da função. O formato do ARN da função deve ser `arn:aws:iam::111122223333:role/role-name`.

   ```
   aws eks create-fargate-profile \
       --fargate-profile-name coredns \
       --cluster-name my-cluster \
       --pod-execution-role-arn
    arn:aws:iam::111122223333:role/AmazonEKSFargatePodExecutionRole \
       --selectors namespace=kube-system,labels={k8s-app=kube-dns} \
       --subnets subnet-0000000000000001 subnet-0000000000000002
    subnet-0000000000000003
   ```

2. Execute e o comando a seguir para remover a anotação padrão `eks.amazonaws.com/compute-type : ec2` dos Pods do CoreDNS.

   ```
   kubectl patch deployment coredns \
       -n kube-system \
       --type json \
       -p='[{"op": "remove", "path": "/spec/template/metadata/annotations/
   eks.amazonaws.com~1compute-type"}]'
   ```

Próximas etapas

- É possível começar a migrar as aplicações existentes para serem executadas no Fargate com o fluxo de trabalho a seguir.

1. Crie um perfil do Fargate que corresponda o namespace do Kubernetes da aplicação aos rótulos do Kubernetes.
2. Exclua e recrie os Pods existentes para que eles sejam agendados no Fargate. Por exemplo, o comando a seguir aciona uma implantação de `coredns`. É possível modificar o namespace e o tipo de implantação para atualizar os Pods específicos.

```
kubectl rollout restart -n kube-system deployment coredns
```

- Implante o Aplicação de roteamento e tráfego HTTP com Application Load Balancers para permitir objetos de Ingresso para os Pods em execução no Fargate.
- É possível usar Ajustar os recursos de pods com o Vertical Pod Autoscaler para definir o tamanho correto da CPU e da memória para os Pods do Fargate e, em seguida, usar Escalar as implantações de pods com o Horizontal Pod Autoscaler para dimensionar a escala desses Pods. Se quiser que o Vertical Pod Autoscaler reimplante automaticamente os Pods no Fargate com combinações maiores de CPU e memória, defina o modo do Vertical Pod Autoscaler como `Auto` ou `Recreate`. Isso ocorre para garantir a funcionalidade correta. Para obter mais informações, consulte a documentação do Vertical Pod Autoscaler no GitHub.
- Você pode configurar o coletor AWS Distro for OpenTelemetry (ADOT) para monitoramento de aplicações seguindo estas instruções.

Defina quais Pods usarão o AWS Fargate quando em execução

> **Important**
>
> O AWS Fargate com o Amazon EKS não está disponível para a AWS GovCloud (Leste dos EUA) e AWS GovCloud (Oeste dos EUA).

Antes de agendar Pods no Fargate no cluster, você deve definir pelo menos um perfil do Fargate que especifique quais Pods usam o Fargate quando iniciados.

Como administrador, é possível usar um perfil do Fargate para declarar quais Pods devem ser executados no Fargate. É possível fazer isso por meio dos seletores de perfil. É possível adicionar até cinco seletores a cada perfil. Cada seletor deve conter um namespace. O seletor também pode incluir rótulos. O campo de rótulo consiste em vários pares de chave-valor opcionais. Os pods que correspondem aos seletores são programados no Fargate. Os pods são correspondidos usando um

namespace e os rótulos que são especificados no seletor. Se um seletor de namespace for definido sem rótulos, o Amazon EKS tenta agendar todos os Pods que são executados nesse namespace para Fargate usando o perfil. Se um Pod a ser agendado corresponder a qualquer um dos seletores no perfil do Fargate, esse Pod será agendado no Fargate.

Se um Pod corresponder a vários perfis do Fargate, será possível especificar qual perfil um Pod usará adicionando o seguinte rótulo de Kubernetes para a especificação do Pod: `eks.amazonaws.com/fargate-profile: my-fargate-profile`. O Pod deve corresponder a um seletor nesse perfil para ser agendado no Fargate. As regras de afinidade e antiafinidade do Kubernetes não são levadas em consideração e não são necessárias com Pods do Amazon EKS Fargate.

Ao criar um perfil do Fargate, é necessário especificar uma função de execução do Pod. Essa função de execução é para os componentes do Amazon EKS que são executados na infraestrutura do Fargate usando o perfil. Ela é adicionada ao Controle de acesso com base em função (RBAC) do Kubernetes do cluster para autorização. Dessa forma, o `kubelet` que está sendo executado na infraestrutura do Fargate pode ser registrado no cluster do Amazon EKS e aparede no cluster como um nó. A função de execução de Pod também fornece permissões do IAM para a infraestrutura do Fargate para permitir acesso de leitura aos repositórios de imagens do Amazon ECR. Para obter mais informações, consulte Perfil do IAM para execução de Pod do Amazon EKS.

Os perfis do Fargate não podem ser alterados. No entanto, é possível criar um perfil atualizado para substituir um perfil existente e excluir o original.

> **Note**
>
> Qualquer Pods que esteja sendo executado usando um perfil do Fargate será interrompido e colocado como pendente quando o perfil for excluído.

Se algum perfil do Fargate em um cluster estiver no status `DELETING`, será necessário aguardar até que a exclusão do perfil do Fargate seja concluída para poder criar qualquer outro perfil nesse cluster.

O Amazon EKS e o Fargate tentam distribuir Pods por todas as sub-redes definidas no perfil do Fargate. No entanto, é possível acabar com uma distribuição desigual. Se você precisar ter uma distribuição uniforme, use dois perfis do Fargate. A distribuição uniforme é importante em cenários em que você deseja implantar duas réplicas e não quer nenhum tempo de inatividade. Recomendamos que cada perfil tenha apenas uma sub-rede.

Componentes do perfil do Fargate

Os componentes a seguir estão contidos em um perfil do Fargate.

Função de execução de pod

Quando o cluster cria Pods no AWS Fargate, o `kubelet` sendo executado na infraestrutura do Fargate precisa fazer as chamadas para as APIs da AWS em seu nome. Por exemplo, ele precisa fazer chamadas para extrair imagens de contêiner do Amazon ECR. A função de execução de Pod do Amazon EKS fornece as permissões do IAM para isso.

Ao criar um perfil do Fargate, é necessário especificar uma função de execução de Pod para ser usada com os Pods. Essa função é adicionada ao Kubernetes Controle de acesso com base em função (RBAC) do cluster para autorização. Isso permite que o `kubelet` que está sendo executado na infraestrutura do Fargate possa ser registrado no cluster do Amazon EKS e apareça no cluster como um nó. Para obter mais informações, consulte Perfil do IAM para execução de Pod do Amazon EKS.

Sub-redes

Os IDs das sub-redes nas quais iniciar Pods que usam esse perfil. No momento, não são atribuídos endereços IP públicos aos Pods sendo executados no Fargate. Portanto, somente sub-redes privadas (sem rota direta para um gateway da Internet) são aceitas para esse parâmetro.

Seletores

Os seletores para encontrar os Pods correspondentes para usar esse perfil do Fargate. É possível especificar até cinco seletores em um perfil do Fargate. Os seletores têm os seguintes componentes:

- Namespace: é necessário especificar um namespace para um seletor. O seletor corresponde apenas a Pods que são criados nesse namespace. No entanto, é possível criar vários seletores para visar a vários namespaces.
- Rótulos: se preferir, você poderá especificar rótulos do Kubernetes para corresponder ao seletor. O seletor só busca correspondências com Pods que tenham todos os rótulos especificados no seletor.

Curingas do perfil do Fargate

Além dos caracteres permitidos pelo Kubernetes, você está autorizado a usar ***** e **?** nos critérios do seletor para namespaces, chaves de rótulos e valores de rótulos:

- **`*`** representa nenhum, um ou vários caracteres. Por exemplo, **`prod*`** pode representar `prod` e `prod-metrics`.
- **`?`** representa um único caractere (por exemplo, **`value?`** pode representar `valuea`). Porém, não pode representar `value` e `value-a`, porque ? só pode representar exatamente um único caractere.

Esses caracteres curinga podem ser usados em qualquer posição e em combinação (por exemplo, **`prod*`**, **`*dev`** e **`frontend*?`**). Outros curingas e outras formas de correspondência de padrões, como expressões regulares, não são compatíveis.

Se houver vários perfis correspondentes para o namespace e os rótulos na especificação do Pod, o Fargate selecionará o perfil com base na classificação alfanumérica pelo nome do perfil. Por exemplo, se ambos os perfis A (com o nome `beta-workload`) e B (com o nome `prod-workload`) tiverem seletores correspondentes para os Pods a serem lançados, o Fargate escolherá o perfil A (`beta-workload`) para os Pods. Os Pods têm rótulos com o perfil A nos Pods (por exemplo, `eks.amazonaws.com/fargate-profile=beta-workload`).

Se você quiser migrar os Pods existentes do Fargate para novos perfis que usem curingas, há duas maneiras de fazer isso:

- Crie um novo perfil com seletores correspondentes e exclua os perfis antigos. Os pods rotulados com perfis antigos serão reprogramados para novos perfis correspondentes.
- Se você quiser migrar workloads, mas não tiver certeza de quais rótulos do Fargate estão em cada Pod do Fargate, será possível usar o método a seguir. Crie um novo perfil com um nome que seja classificado em ordem alfanumérica primeiro entre os perfis no mesmo cluster. Em seguida, recicle os Pods do Fargate que precisem ser migrados para novos perfis.

Criar um perfil do Fargate

Esta seção descreve como criar um perfil do Fargate. Você também deve ter criado uma função de execução de Pod a ser usada para o perfil do Fargate. Para obter mais informações, consulte Perfil do IAM para execução de Pod do Amazon EKS. Pods em execução no Fargate apenas compatíveis apenas em sub-redes privadas com acesso via gateway NAT a Serviços da AWS, mas não com uma rota direta para um gateway da Internet. Isso ocorre para que a VPC do seu cluster tenha sub-redes privadas disponíveis. Você pode criar um perfil com o `eksctl` ou o AWS Management Console.

eksctl

Para criar um perfil do Fargate com o **`eksctl`**

Crie o perfil do Fargate com o comando `eksctl` a seguir, substituindo cada *`example value`* por seus próprios valores. Você precisa especificar um namespace. Mas, a opção `--labels` não é exigida.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

É possível usar certos curingas para rótulos *`my-kubernetes-namespace`* e *`key=value`* . Para obter mais informações, consulte Curingas do perfil do Fargate.

AWS Management Console

Como criar um perfil do Fargate para um cluster com o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha o cluster para o qual deve ser criado um perfil do Fargate.
3. Escolha a guia Compute (Computação).
4. Em Fargate profile (Perfil do Fargate), escolha Add Fargate profile (Adicionar perfil do Fargate).
5. Na página Configure Fargate profile (Configurar o perfil do Fargate), faça o seguinte:
    a. Em Name (Nome), insira um nome exclusivo para o perfil do Fargate, como ***`my-profile`***.
    b. Em Pod execution role (Função de execução de pod), escolha a função de execução de Pod que deve ser usada com o perfil do Fargate. Somente funções do IAM com o principal de serviço do `eks-fargate-pods.amazonaws.com` são mostradas. Se nenhuma função estiver listada, você deve criar uma. Para obter mais informações, consulte Perfil do IAM para execução de Pod do Amazon EKS.
    c. Modifique as sub-redes selecionadas conforme necessário.

> **Note**
>
> Somente sub-redes privadas são compatíeis com Pods em execução no Fargate.

d. Em Tags (Etiquetas), você tem a opção de etiquetar o perfil do Fargate. Essas tags não são propagadas para outros recursos associados ao perfil, como Pods.

e. Escolha Próximo.

6. Na página Configure Pod selection (Configurar a seleção de), faça o seguinte:

a. Em Namespace, insira um namespace para fazer a correspondência com Pods.

- É possível usar namespaces específicos para a correspondência, como **`kube-system`** ou **`default`**.
- É possível usar certos curingas (por exemplo, **`prod-*`**) para fazer a correspondência com vários namespaces (por exemplo, `prod-deployment` e `prod-test`). Para obter mais informações, consulte Curingas do perfil do Fargate.

b. (Opcional) Adicione rótulos Kubernetes ao seletor. Adicione-os especificamente ao seletor ao qual os Pods no namespace especificado precisam corresponder.

- É possível adicionar o rótulo **`infrastructure: fargate`** ao seletor para que somente Pods no namespace especificado que também tenham o rótulo Kubernetes do `infrastructure: fargate` correspondam ao seletor.
- É possível usar certos curingas (por exemplo, **`key?: value?`**) para fazer a correspondência com vários namespaces (por exemplo, `keya: valuea` e `keyb: valueb`). Para obter mais informações, consulte Curingas do perfil do Fargate.

c. Escolha Próximo.

7. Na página Review and create (Revisar e criar), revise as informações do perfil do Fargate e selecione Create (Criar).

Excluir um perfil do Fargate

Este tópico descreve como excluir um perfil do Fargate. Quando você exclui um perfil do Fargate, todos os Pods que foram agendados no Fargate com o perfil são excluídos. Se esses Pods corresponderem a outro perfil do Fargate, eles serão agendados no Fargate com esse perfil. Se eles

não corresponderem mais a nenhum perfil do Fargate, eles não estão agendados para o Fargate e poderão continuar pendentes.

Somente um perfil do Fargate em um cluster pode estar no status `DELETING` de cada vez. Espere que um perfil do Fargate conclua a exclusão antes de excluir qualquer outro perfil desse cluster.

Você pode excluir um perfil com o `eksctl`, o AWS Management Console ou a AWS CLI. Selecione a guia com o nome da ferramenta que você deseja usar para excluir o perfil.

eksctl

Para excluir um perfil do Fargate com **`eksctl`**

Use o comando a seguir para excluir um perfil de um cluster. Substitua cada *`example value`* por seus próprios valores.

```
eksctl delete fargateprofile  --name my-profile --cluster my-cluster
```

AWS Management Console

Como excluir um perfil do Fargate de um cluster com o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, escolha Clusters. Na lista de clusters, escolha o cluster do qual deseja excluir o perfil do Fargate.
3. Escolha a guia Compute (Computação).
4. Escolha o perfil do Fargate a ser excluído e, em seguida, escolha Delete (Excluir).
5. Na página Delete Fargate Profile (Excluir perfil do Fargate), insira o nome do perfil e, em seguida, escolha Delete (Excluir).

AWS CLI

Para excluir um perfil do Fargate com AWS CLI

Use o comando a seguir para excluir um perfil de um cluster. Substitua cada *`example value`* por seus próprios valores.

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-cluster
```

Entender os detalhes de configuração do Pod do Fargate

> **Important**
>
> O AWS Fargate com o Amazon EKS não está disponível para a AWS GovCloud (Leste dos EUA) e AWS GovCloud (Oeste dos EUA).

Esta seção descreve alguns dos detalhes exclusivos da configuração de Pod para execução de Kubernetes do Pods no AWS Fargate.

CPU e memória do Pod

Com o Kubernetes, é possível definir solicitações, uma quantidade mínima de vCPU e recursos de memória que são alocados para cada contêiner em um Pod. Os Pods são programados pelo Kubernetes para garantir que ao menos os recursos solicitados para cada Pod estejam disponíveis no recurso computacional. Para obter mais informações, consulte Gerenciar recursos de computação para contêiners, na documentação do Kubernetes.

> **Note**
>
> Como o Amazon EKS Fargate executa apenas um Pod por nó, não ocorre o despejo de Pods em caso de menos recursos. Todos os Pods do Amazon EKS Fargate são executados com prioridade garantida, então a CPU e a memória solicitadas devem ser iguais ao limite para todos os contêineres. Para obter mais informações, consulte Configurar a qualidade do serviço para Pods na documentação do Kubernetes.

Quando os Pods são agendados no Fargate, as reservas de vCPU e de memória dentro da especificação do Pod determinam a quantidade de CPU e de memória que deve ser provisionada para o Pod.

- A solicitação máxima de qualquer contêiner de inicialização é usada para determinar os requisitos de vCPU e de memória da solicitação de inicialização.
- As solicitações para todos os contêineres de longa execução são adicionadas para determinar os requisitos de vCPU e de memória da solicitação de longa duração.
- O maior dos dois valores anteriores é escolhido para que a solicitação de vCPU e de memória use em seu Pod.

- O Fargate adiciona 256 MB a cada reserva de memória do Pod para os componentes exigidos do Kubernetes (`kubelet`, `kube-proxy` e `containerd`).

O Fargate faz o arredondamento para a seguinte configuração de computação que mais se aproxima da soma das solicitações de vCPU e de memória, a fim de garantir que os Pods sempre tenham os recursos necessários para execução.

Se você não especificar uma combinação de vCPU e de memória, será usada a menor combinação (0,25 de vCPU e 0,5 GB de memória).

A tabela a seguir mostra as combinações de vCPU e de memória que estão disponíveis para Pods em execução no Fargate.

Valor de vCPU	Valor de memória
0,25 vCPU	0,5 GB, 1 GB, 2 GB
0,5 vCPU	1 GB, 2 GB, 3 GB, 4 GB
1 vCPU	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB
2 vCPU	Entre 4 GB e 16 GB em incrementos de 1 GB
4 vCPU	Entre 8 GB e 30 GB em incrementos de 1 GB
8 vCPU	Entre 16 GB e 60 GB em incrementos de 4 GB
16 vCPU	Entre 32 GB e 120 GB em incrementos de 8 GB

A memória adicional reservada para os componentes do Kubernetes pode fazer com que uma tarefa do Fargate com mais vCPUs do que o solicitado seja provisionada. Por exemplo, uma solicitação de 1 vCPU e 8 GB de memória terá 256 MB adicionados à solicitação de memória e provisionará uma tarefa do Fargate com 2 vCPUs e 9 GB de memória, já que nenhuma tarefa com 1 vCPU e 9 GB de memória está disponível.

Não há correlação entre o tamanho do Pod em execução no Fargate e o tamanho do nó informado pelo Kubernetes com `kubectl get nodes`. O tamanho do nó informado geralmente é maior

do que a capacidade do Pod. Você pode verificar a capacidade do Pod com o comando a seguir. Substitua *default* pelo namespaces do Pod e *pod-name* pelo nome do seu Pod.

```
kubectl describe pod --namespace default pod-name
```

Veja um exemplo de saída abaixo.

```
[...]
annotations:
    CapacityProvisioned: 0.25vCPU 0.5GB
[...]
```

A anotação de `CapacityProvisioned` representa a capacidade imposta do Pod e determina o custo do Pod em execução no Fargate. Para obter informações sobre preços das configurações de computação, consulte Preços do AWS Fargate Fargate.

Armazenamento do Fargate

Um Pod em execução no Fargate monta automaticamente um sistema de arquivos do Amazon EFS. Você não pode usar o provisionamento dinâmico de volume persistente com nós do Fargate, mas pode usar o provisionamento estático. Para obter mais informações, consulte Driver da CSI do Amazon EFS no GitHub.

Quando provisionado, cada Pod executado no Fargate recebe um padrão de 20 GiB de armazenamento temporário. Esse tipo de armazenamento é excluído depois que um Pod é interrompido. Os novos Pods lançados no Fargate têm criptografia de volume de armazenamento temporário habilitada por padrão. O armazenamento temporário do Pod é criptografado com um algoritmo de criptografia AES-256 usando chaves gerenciadas pelo AWS Fargate.

> **Note**
>
> O armazenamento utilizável padrão para Pods do Amazon EKS executados no Fargate é inferior a 20 GiB. Isso ocorre porque algum espaço é usado pelo `kubelet` e outros módulos Kubernetes que são carregados no Pod.

É possível aumentar a quantidade total de armazenamento temporário para até 175 GiB. Para configurar o tamanho com o Kubernetes, especifique as solicitações do recurso `ephemeral-`

`storage` para cada contêiner em um Pod. Quando o Kubernetes programar Pods, garanta que a soma das solicitações de recursos para cada Pod seja menor que a capacidade da tarefa do Fargate. Para obter mais informações, consulte Gerenciamento de recursos para Pods e contêiners na documentação do Kubernetes.

O Amazon EKS Fargate fornece mais armazenamento temporário do que o solicitado para fins de uso do sistema. Por exemplo, uma solicitação de 100 GiB provisionará uma tarefa do Fargate com armazenamento temporário de 115 GiB.

Definir ações para eventos de correção do sistema operacional do AWS Fargate

> **Important**
>
> O AWS Fargate com o Amazon EKS não está disponível para a AWS GovCloud (Leste dos EUA) e AWS GovCloud (Oeste dos EUA).

O Amazon EKS corrige periodicamente o sistema operacional para AWS Fargate nós para mantê-los seguros. Como parte do processo de correção, reciclamos os nós para instalar os patches do sistema operacional. Há tentativas de atualizações de uma forma que gera o menor impacto possível nos seus serviços. Porém, se os Pods não forem removidos com êxito, haverá momentos em que eles deverão ser excluídos. As ações a seguir podem ser executadas para minimizar possíveis interrupções:

- Defina orçamentos de interrupção de Pod (PDBs) apropriados para controlar o número de Pods que se tornam inativos ao mesmo tempo.
- Crie regras de do Amazon EventBridge para reagir a remoções com falha antes que os Pods sejam excluídos.
- Crie uma configuração de notificação em Notificações de Usuários da AWS.

O Amazon EKS trabalha em conjunto com a comunidade Kubernetes para disponibilizar correções de bugs e patches de segurança da maneira mais rápida possível. Todos os Pods do Fargate começam com a versão de patch do Kubernetes mais recente, disponível no Amazon EKS para a versão do Kubernetes do cluster. Se você tiver um Pod com uma versão de patch mais antiga, o Amazon EKS poderá reciclar para atualizá-lo para a versão mais recente. Isso garante que os Pods tenham as atualizações de segurança mais recentes. Assim, se houver um problema crítico de

Vulnerabilidades e exposições comuns (CVE), você permanecerá atualizado para reduzir os riscos de segurança.

Para limitar o número de Pods inativos ao mesmo tempo quando os Pods são reciclados, é possível definir orçamentos de interrupção de Pod (PDBs). É possível usar PDBs para definir a disponibilidade mínima com base nos requisitos de cada uma das suas aplicações e, ao mesmo tempo, possibilitar que as atualizações ocorram. Para saber mais, consulte Specifying a Disruption Budget for your Application (Especificar um orçamento de interrupção para a aplicação), na Documentação do Kubernetes.

O Amazon EKS utiliza a API de remoção para drenar o Pod com segurança, respeitando ao mesmo tempo os PDBs definidos para a aplicação. Os pods são removidos por zona de disponibilidade para minimizar os impactos. Se a remoção tiver êxito, o novo Pod receberá o patch mais recente, e nenhuma ação adicional será necessária.

Quando a remoção de um Pod falha, o Amazon EKS envia um evento à sua conta com detalhes sobre os Pods cuja remoção falhou. É possível atuar na mensagem antes do horário de encerramento programado. O tempo específico varia dependendo da urgência do patch. Quando chega a hora, o Amazon EKS tenta remover os Pods novamente. Porém, desta vez, um novo evento não será enviado se a remoção falhar. Se a remoção falhar novamente, os Pods existentes serão excluídos periodicamente, para que novos Pods possam receber o patch mais recente.

Veja a seguir uma amostra de evento recebido quando a remoção do Pod falha. Ele inclui detalhes sobre o cluster, o nome do Pod, o namespace do Pod, o perfil do Fargate e o horário de término agendado.


```
{
    "version": "0",
    "id": "12345678-90ab-cdef-0123-4567890abcde",
    "detail-type": "EKS Fargate Pod Scheduled Termination",
    "source": "aws.eks",
    "account": "111122223333",
    "time": "2021-06-27T12:52:44Z",
    "region": "region-code",
    "resources": [
        "default/my-database-deployment"
    ],
    "detail": {
        "clusterName": "my-cluster",
        "fargateProfileName": "my-fargate-profile",
```

```
        "podName": "my-pod-name",
        "podNamespace": "default",
        "evictErrorMessage": "Cannot evict pod as it would violate the pod's disruption
 budget",
        "scheduledTerminationTime": "2021-06-30T12:52:44.832Z[UTC]"
    }
}
```

Além disso, a existência de vários PDBs associados a um Pod pode causar um evento de falha de remoção. Esse evento retorna a mensagem de erro a seguir.

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction
 subresource does not support",
```

É possível criar uma ação desejada com base nesse evento. Por exemplo, é possível ajustar o orçamento de interrupção de Pod (PDB) para controlar como os Pods são removidos. De maneira mais específica, suponha que você comece com um PDB que define a porcentagem-alvo de Pods disponíveis. Antes que os Pods sejam encerrados à força durante um upgrade, é possível ajustar o PDB para uma porcentagem diferente de Pods. Para receber esse evento, você deve criar uma regra do Amazon EventBridge no Conta da AWS e no Região da AWS ao qual o cluster pertence. A regra deve utilizar o seguinte Padrão personalizado: Para obter mais informações, consulte Criar regras do Amazon EventBridge que reajam a eventos no Guia do usuário do Amazon EventBridge.

```
{
  "source": ["aws.eks"],
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

Um destino apropriado pode ser definido para o evento capturá-lo. Para acessar uma lista completa de destinos disponíveis, consulte Destinos do Amazon EventBridge, no Guia do usuário do Amazon EventBridge. Também é possível criar uma configuração de notificação em Notificações de Usuários da AWS. Ao usar o AWS Management Console para criar a notificação, em Regras de eventos, escolha Elastic Kubernetes Service (EKS) como Nome do Serviço da AWS e EKS Fargate Pod Scheduled Termination para Tipo de evento. Para obter mais informações, consulte Introdução às Notificações de Usuários da AWS no Guia do usuário das Notificações de Usuários da AWS.

Coletar métricas de aplicações e uso do AWS Fargate

> **Important**
>
> O AWS Fargate com o Amazon EKS não está disponível para a AWS GovCloud (Leste dos EUA) e AWS GovCloud (Oeste dos EUA).

É possível coletar métricas do sistema e métricas de uso do CloudWatch para o AWS Fargate.

Métricas da aplicação

Para aplicações em execução no Amazon EKS e no AWS Fargate, você pode usar o AWS Distro for OpenTelemetry (ADOT). O ADOT permite coletar métricas do sistema e enviá-las para os painéis do CloudWatch Container Insights. Para começar a usar o ADOT para aplicações em execução no Fargate, consulte Usar o CloudWatch Container Insights com o Distro AWS para OpenTelemetry, na documentação do ADOT.

Métricas de uso

É possível usar métricas de uso do CloudWatch para fornecer visibilidade sobre o uso de recursos de sua conta. Use essas métricas para visualizar o uso do serviço atual nos gráficos e painéis do CloudWatch.

As métricas de uso do AWS Fargate correspondem às cotas de serviço da AWS. Também é possível configurar alarmes que alertem você quando o uso se aproximar de uma cota de serviço. Para obter mais informações sobre cotas de serviço do Fargate, consulte Visualizar e gerenciar o Amazon EKS e as cotas de serviço do Fargate.

O AWS Fargate publica as seguintes métricas no namespace `AWS/Usage`.

Métrica	Descrição
`ResourceCount`	O número total dos recursos especificados em execução na sua conta. Os recursos são definidos pelas dimensões associadas à métrica.

As dimensões a seguir são usadas para refinar as métricas de uso publicadas pelo AWS Fargate.

Dimensão	Descrição
`Service`	O nome do serviço da AWS que contém o recurso. Para as métricas de uso do AWS Fargate, o valor dessa dimensão é `Fargate`.
`Type`	O tipo de entidade que está sendo relatada. Atualmente, o único valor válido para métricas de uso do AWS Fargate é `Resource`.
`Resource`	O tipo de recurso que está em execução. Atualmente, o AWS Fargate retorna informações sobre o uso sob demanda do Fargate. O valor do recurso para uso sob demanda do Fargate é `OnDemand`. **Note** O uso sob demanda do Fargate combina Pods do Amazon EKS usando Fargate, tarefas do Amazon ECS usando o tipo de inicialização do Fargate e tarefas do Amazon ECS usando o provedor de capacidade do `FARGATE`.
`Class`	A classe do recurso sob acompanhamento. Atualmente, o AWS Fargate não usa a dimensão de classe.

Criar um alarme do CloudWatch para monitorar as métricas de uso dos recursos do Fargate

O AWS Fargate fornece as métricas de uso do CloudWatch que correspondem às cotas de serviço da AWS para uso de recursos sob demanda do Fargate. No console do Service Quotas, você pode visualizar seu uso em um gráfico. Também é possível configurar alarmes que alertam você quando o uso se aproxima de uma cota de serviço. Para obter mais informações, consulte Coletar métricas de aplicações e uso do AWS Fargate.

Use as etapas a seguir para criar um alarme do CloudWatch com base em uma das métricas de uso dos recursos do Fargate.

Como criar um alarme com base nas cotas de uso do Fargate (AWS Management Console)

1. Abra o console Service Quotas em https://console.aws.amazon.com/servicequotas/.
2. No painel de navegação à esquerda, escolha AWS services (Serviços da).
3. Na lista de serviços da AWS, procure e selecione AWS Fargate.
4. Na lista Service quotas (Cotas de serviço), escolha a cota de uso do Fargate para a qual você deseja criar um alarme.
5. Na seção “Amazon CloudWatch alarms” (Alarmes do Amazon CloudWatch), escolha Create (Criar).
6. Em Alarm threshold (Limitação do alarme), escolha a porcentagem do valor da cota aplicada que você deseja definir como o valor do alarme.
7. Em Alarm name (Nome do alarme), insira um nome para o alarme e selecione Create (Criar).

Iniciar o log do AWS Fargate para seu cluster

> **Important**
>
> O AWS Fargate com o Amazon EKS não está disponível para a AWS GovCloud (Leste dos EUA) e AWS GovCloud (Oeste dos EUA).

O Amazon EKS no Fargate oferece um roteador de log integrado baseado no Fluent Bit. Isso significa que você não executa explicitamente um contêiner do Fluent Bit separadamente, mas a Amazon o executa para você. Tudo o que você precisa fazer é configurar o roteador de log. A configuração acontece por meio de um `ConfigMap` que deve atender aos seguintes critérios:

- Nomeado `aws-logging`
- Criado em um namespace dedicado chamado `aws-observability`
- Não pode exceder 5.300 caracteres.

Depois de criar o `ConfigMap`, o Amazon EKS no Fargate o detecta automaticamente e configura o roteador de log com ele. O Fargate usa uma versão do AWS para Fluent Bit, uma distribuição do Fluent Bit upstream compatível e gerenciada pela AWS. Para obter mais informações, consulte AWS para Fluent Bit no GitHub.

O roteador de log permite que você use toda a variedade de serviços da AWS para análise de log e armazenamento. Você pode transmitir logs do Fargate diretamente para o Amazon CloudWatch, Amazon OpenSearch Service. Você também pode transmitir logs para destinos como Amazon S3, Amazon Kinesis Data Streams e ferramentas de parceiros por meio do Amazon Data Firehose.

Pré-requisitos

- Um perfil existente do Fargate que especifica um namespace existente do Kubernetes no qual você implanta Pods do Fargate. Para obter mais informações, consulte Etapa 3: criar um perfil do Fargate para o cluster.
- Um perfil de execução de Pod do Fargate existente. Para obter mais informações, consulte Etapa 2: criar um perfil de execução de Pod do Fargate.

Configuração do roteador de log

Para configurar o roteador de log

Nas etapas a seguir, substitua todos os *example value* por seus próprios valores.

1. Criar um namespace Kubernetes dedicado com o nome `aws-observability`.

   a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado *aws-observability-namespace*`.yaml`: O valor para `name` deve ser `aws-observability` e o rótulo `aws-observability: enabled` é obrigatório.

   ```
   kind: Namespace
   apiVersion: v1
   metadata:
     name: aws-observability
     labels:
       aws-observability: enabled
   ```

   b. Crie o novo namespace.

   ```
   kubectl apply -f aws-observability-namespace.yaml
   ```

2. Crie um `ConfigMap` com um valor de dados do `Fluent Conf` para enviar logs de contêiner a um destino. Fluent Conf é o Fluent Bit, uma linguagem de configuração de processador de log rápida e leve usada para encaminhar logs de contêiner para um destino de sua escolha. Para obter mais informações, consulte Arquivo de configuração, na documentação do Fluent Bit.

> **Important**
>
> As principais seções incluídas em um `Fluent Conf` típico são `Service`, `Input`, `Filter` e `Output`. No entanto, o roteador de log do Fargate só aceita:
>
> - As seções `Filter` e `Output`.
> - Uma seção `Parser`.
>
> Se você fornecer outras seções, elas serão rejeitadas.

O roteador de log do Fargate gerencia as seções `Input` e `Service`. Ele tem a seção `Input` a seguir, que não pode ser modificada e não é necessária no `ConfigMap`. Porém, ela pode oferecer informações, como o limite do buffer de memória e a tag aplicada aos logs.

```
[INPUT]
    Name tail
    Buffer_Max_Size 66KB
    DB /var/log/flb_kube.db
    Mem_Buf_Limit 45MB
    Path /var/log/containers/*.log
    Read_From_Head On
    Refresh_Interval 10
    Rotate_Wait 30
    Skip_Long_Lines On
    Tag kube.*
```

Ao criar o `ConfigMap`, leve em consideração as seguintes regras que o Fargate usa para validar os campos:

- `[FILTER]`,`[OUTPUT]`, e `[PARSER]` devem ser especificados sob cada chave correspondente. Por exemplo, o `[FILTER]` deve estar em `filters.conf`. Você pode ter um ou mais `[FILTER]`s em `filters.conf`. As seções `[OUTPUT]` e `[PARSER]` também devem estar sob suas chaves correspondentes. Ao especificar várias seções `[OUTPUT]`, você pode rotear os logs para destinos diferentes ao mesmo tempo.

- O Fargate valida as chaves necessárias para cada seção. `Name` e `match` são necessários para cada `[FILTER]` e `[OUTPUT]`. `Name` e `format` são necessários para cada `[PARSER]`. As chaves não diferenciam maiúsculas de minúsculas.
- Variáveis de ambiente como `${ENV_VAR}` não são permitidas no `ConfigMap`.
- O recuo tem que ser o mesmo para o par de diretivas ou chave-valor dentro de cada `filters.conf`, `output.conf` e `parsers.conf`. Pares chave-valor têm que ser recuados mais do que as diretivas.
- O Fargate valida de acordo com os seguintes filtros compatíveis: `grep`, `parser`, `record_modifier`, `rewrite_tag`, `throttle`, `nest`, `modify` e `kubernetes`.
- O Fargate valida de acordo com os seguintes filtros compatíveis: `es`, `firehose`, `kinesis_firehose`, `cloudwatch`, `cloudwatch_logs` e `kinesis`.
- É necessário fornecer pelo menos um plugin `Output` com suporte no `ConfigMap` para habilitar o registro. `Filter` e `Parser` não são necessários para habilitar o registro.

Você também pode executar o Fluent Bit no Amazon EC2 usando a configuração desejada para solucionar quaisquer problemas decorrentes da validação. Crie o `ConfigMap` usando um dos exemplos a seguir.

> **Important**
>
> O registro em log do Fargate no Amazon EKS não oferece suporte para a configuração dinâmica de `ConfigMaps`. Quaisquer alterações nos `ConfigMaps` são aplicadas somente a novos Pods. As alterações não são aplicadas a Pods existentes.

Crie um `ConfigMap` usando o exemplo para o destino de log desejado.

> **Note**
>
> Você também pode usar o Amazon Kinesis Data Streams para seu destino de log. Se você usar o Kinesis Data Streams, verifique se a função de execução do pod recebeu a permissão `kinesis:PutRecords`. Para obter mais informações, consulte Definição de preço do Amazon Kinesis Data Streams em Fluent Bit: Manual oficial.

CloudWatch

Para criar um **`ConfigMap`** para o CloudWatch

Você tem duas opções de saída ao usar o CloudWatch:

- Um plugin de saída criado em C
- Um plugin de saída criado em Golang

O exemplo a seguir mostra como usar o plug-in `cloudwatch_logs` para enviar logs para o CloudWatch.

1. Salve o conteúdo a seguir em um arquivo denominado *`aws-logging-cloudwatch-configmap`*`.yaml`. Substitua *`region-code`* pela Região da AWS em que está o cluster. Os parâmetros em `[OUTPUT]` são necessários.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  flb_log_cw: "false"  # Set to true to ship Fluent Bit process logs to
 CloudWatch.
  filters.conf: |
    [FILTER]
        Name parser
        Match *
        Key_name log
        Parser crio
    [FILTER]
        Name kubernetes
        Match kube.*
        Merge_Log On
        Keep_Log Off
        Buffer_Size 0
        Kube_Meta_Cache_TTL 300s
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
```

```
        Match   kube.*
        region region-code
        log_group_name my-logs
        log_stream_prefix from-fluent-bit-
        log_retention_days 60
        auto_create_group true
  parsers.conf: |
    [PARSER]
        Name crio
        Format Regex
        Regex ^(?<time>[^ ]+) (?<stream>stdout|stderr) (?<logtag>P|F) (?
<log>.*)$
        Time_Key    time
        Time_Format %Y-%m-%dT%H:%M:%S.%L%z
```

2. Aplique o manifesto ao cluster.

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

3. Baixe a política de IAM do CloudWatch no computador. Você também pode visualizar a política no GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/cloudwatchlogs/permissions.json
```

Amazon OpenSearch Service

Para criar um **ConfigMap** para o Amazon OpenSearch Service

Se você quiser enviar logs ao Amazon OpenSearch Service, pode utilizar a saída es, que é um plugin escrito em C. O exemplo a seguir mostra como usar o plug-in para enviar logs para o OpenSearch.

1. Salve o conteúdo a seguir em um arquivo denominado *aws-logging-opensearch-configmap*.yaml. Substitua cada *example value* por seus próprios valores.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
```

```
data:
  output.conf: |
    [OUTPUT]
      Name  es
      Match *
      Host  search-example-gjxdcilagiprbglqn42jsty66y.region-
code.es.amazonaws.com
      Port  443
      Index example
      Type  example_type
      AWS_Auth On
      AWS_Region region-code
      tls   On
```

2. Aplique o manifesto ao cluster.

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

3. Baixe a política do IAM OpenSearch para o seu computador. Você também pode visualizar a política no GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-
fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/
permissions.json
```

   Verifique se o controle de acesso do OpenSearch Dashboards está configurado corretamente. O `all_access role` no OpenSearch Dashboards necessita que o perfil de execução de Pod do Fargate e o perfil do IAM sejam mapeados. O mesmo mapeamento deve ser feito para a função `security_manager`. Você pode adicionar os mapeamentos anteriores selecionando `Menu`, depois `Security`, depois `Roles` e, em seguida, selecionar as respectivas funções. Para obter mais informações, consulte How do I troubleshoot CloudWatch Logs so that it streams to my Amazon ES domain? (Como solucionar problemas do CloudWatch Logs para que ele transmita para o meu domínio do Amazon ES?).

Firehose

Para criar um **`ConfigMap`** para o Firehose

Você tem duas opções de saída ao enviar logs para o Firehose:

- `kinesis_firehose` – Um plugin de saída criado em C.
- `firehose` – Um plugin de saída criado em Golang.

O exemplo a seguir mostra como usar o plug-in `kinesis_firehose` para enviar logs para o Firehose.

1. Salve o conteúdo a seguir em um arquivo denominado *`aws-logging-firehose-configmap`*`.yaml`. Substitua *`region-code`* pela Região da AWS em que está o cluster.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
     Name  kinesis_firehose
     Match *
     region region-code
     delivery_stream my-stream-firehose
```

2. Aplique o manifesto ao cluster.

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. Baixe a política do IAM do Firehose para o seu computador. Você também pode visualizar a política no GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/kinesis-firehose/permissions.json
```

3. Crie uma política do IAM usando ao arquivo de política obtido por download em uma etapa anterior.

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document
 file://permissions.json
```

4. Anexe a política do IAM à função de execução de pod especificada para o seu perfil do Fargate com o comando a seguir. Substitua *`111122223333`* pelo ID da sua conta. Substitua

AmazonEKSFargatePodExecutionRole pelo seu perfil de execução de Pod (para saber mais, consulte Etapa 2: criar um perfil de execução de Pod do Fargate).

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Suporte ao filtro Kubernetes

Esse recurso requer a seguinte versão mínima do Kubernetes e de nível de plataforma, ou versões superiores:

Versão do Kubernetes	Nível da plataforma
1.23 e posteriores	eks.1

O filtro Fluent Bit Kubernetes permite adicionar metadados Kubernetes a arquivos de log. Para obter mais informações sobre o filtro, consulte Kubernetes na documentação do Fluent Bit. É possível aplicar um filtro usando o endpoint do servidor da API.

```
filters.conf: |
    [FILTER]
        Name             kubernetes
        Match            kube.*
        Merge_Log           On
        Buffer_Size         0
        Kube_Meta_Cache_TTL 300s
```

Important

- `Kube_URL`, `Kube_CA_File`, `Kube_Token_Command` e `Kube_Token_File` são parâmetros de configuração de propriedade do serviço e não devem ser especificados. O Amazon EKS Fargate preenche esses valores.
- `Kube_Meta_Cache_TTL` é o tempo em que o Fluent Bit aguarda até se comunicar com o servidor de API para obter os metadados mais recentes. Se `Kube_Meta_Cache_TTL` não

for especificado, o Amazon EKS Fargate acrescentará um valor padrão de 30 minutos para diminuir a carga no servidor de API.

Para enviar logs de processo do Fluent Bit para sua conta

Você pode enviar logs de processo do Fluent Bit para o Amazon CloudWatch usando o `ConfigMap` a seguir. O envio de logs do processo Fluent Bit para o CloudWatch exige custos adicionais de ingestão e armazenamento de logs. Substitua *`region-code`* pela Região da AWS em que está o cluster.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
  labels:
data:
  # Configuration files: server, input, filters and output
  # ======================================================
  flb_log_cw: "true"  # Ships Fluent Bit process logs to CloudWatch.

  output.conf: |
    [OUTPUT]
        Name cloudwatch
        Match kube.*
        region region-code
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
        auto_create_group true
```

Os logs estão na Região da AWS em que o cluster se localiza no CloudWatch. O nome do grupo de logs é *`my-cluster`*`-fluent-bit-logs`, e o nome do fluxo de logs do Fluent Bit é `fluent-bit-`*`podname`*`-`*`pod-namespace`*.

Note

- Os logs de processo são enviados somente quando o processo do Fluent Bit é iniciado corretamente. Caso haja uma falha ao iniciar o Fluent Bit, os logs do processo serão perdidos. Só é possível enviar logs de processo ao CloudWatch.

- Para depurar o envio de logs de processo na sua conta, é possível aplicar o `ConfigMap` anterior para obter os logs de processo. A falha de inicialização do Fluent Bit geralmente ocorre porque seu `ConfigMap` não foi analisado ou aceito pelo Fluent Bit durante a inicialização.

Para interromper os registros do processo Fluent Bit de envio

O envio de logs do processo Fluent Bit para o CloudWatch exige custos adicionais de ingestão e armazenamento de logs. Para excluir registros do processo em uma configuração `ConfigMap` existente, siga as etapas a seguir.

1. Localize o grupo de logs do CloudWatch criado automaticamente para os registros de logs do processo Fluent Bit do cluster do Amazon EKS, após ativar o registro do Fargate. Ele segue o formato `{cluster_name}-fluent-bit-logs`.
2. Exclua os fluxos de log existentes do CloudWatch criados para os logs de processo de cada Pod's no grupo de logs do CloudWatch.
3. Edite `ConfigMap` e configure `flb_log_cw: "false"`.
4. Reinicie todos os Pods existentes no cluster.

Testar a aplicação

1. Implante uma Pod de exemplo.
   a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado *`sample-app`*`.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
  namespace: same-namespace-as-your-fargate-profile
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
```

```
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - name: http
              containerPort: 80
```

b. Aplique o manifesto ao cluster.

```
kubectl apply -f sample-app.yaml
```

2. Visualize os logs do NGINX usando os destinos que você configurou no `ConfigMap`.

Considerações sobre tamanho

Sugerimos que você planeje até 50 MB de memória para o roteador de log. Se você espera que a aplicação gere logs com um throughput muito alto, planeje até 100 MB.

Solução de problemas

Para confirmar se o recurso de registro em log está habilitado ou desabilitado por algum motivo, como um `ConfigMap` inválido, e por que ele é inválido, verifique os eventos do Pod com **`kubectl describe pod`** ***`pod_name`***. A saída pode incluir eventos do Pod que esclarecem se o registro em log está habilitado ou não, como a saída de exemplo a seguir.

```
[...]
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
                      Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
                      kubernetes.io/psp: eks.privileged
[...]
Events:
  Type     Reason           Age        From
              Message
  ----     ------           ----       ----
              -------
  Warning  LoggingDisabled  <unknown>  fargate-scheduler
               Disabled logging because aws-logging configmap was not found. configmap
 "aws-logging" not found
```

Os eventos do Pod são efêmeros, com um intervalo de tempo que depende das configurações. Você também pode visualizar as anotações de um Pod's usando **`kubectl describe pod`** ***`pod-name`***. A anotação do Pod informa se o recurso de registro em log está habilitado ou desabilitado e o motivo.

Escolher um tipo de instância de nó do Amazon EC2 ideal

O Amazon EC2 fornece uma extensa seleção de tipos de instância para nós de processamento. Cada tipo de instância oferece diferentes capacidades de computação, memória, armazenamento e rede. Cada instância também é agrupada em famílias de acordo com esses recursos. Para obter uma lista, consulte Tipos de instâncias disponíveis, no Guia do usuário do Amazon EC2, e Tipos de instâncias disponíveis, no Guia do usuário do Amazon EC2. O Amazon EKS lança diversas variações de AMIs do Amazon EC2 para permitir suporte. Para garantir que o tipo de instância selecionado seja compatível com o Amazon EKS, considere estes critérios.

- Todas as AMIs do Amazon EKS no momento não têm suporte para as famílias `g5g` e `mac`.
- Arm e AMIs do Amazon EKS não aceleradas não têm suporte para as famílias `g3`, `g4`, `inf` e `p`.
- AMIs aceleradas do Amazon EKS não têm suporte para as famílias `a`, `c`, `hpc`, `m` e `t`.
- Para instâncias baseadas em ARM, o Amazon Linux 2023 (AL2023) é somente compatível com tipos de instância que usam processadores Graviton2 ou em versões posteriores. O AL2023 não é compatível com instâncias `A1`.

Ao escolher entre tipos de instância que têm suporte pelo Amazon EKS, considere os recursos a seguir de cada tipo.

Número de instâncias em um grupo de nós

> Em geral, um número menor de instâncias maiores é melhor, especialmente se você tiver muitos Daemonsets. Cada instância requer chamadas de API para o servidor de API, portanto, quanto mais instâncias você tiver, maior a carga no servidor de APIs.

Sistema operacional

> Revise os tipos de instância compatíveis com o Linux, o Windows e o Bottlerocket. Antes de criar instâncias do Windows, revise Implantar nós do Windows em clusters do EKS.

Arquitetura de hardware

Você precisa de x86 ou de Arm? Antes de implantar instâncias do Arm, revise AMIs Amazon Linux Arm otimizadas para Amazon EKS. Você precisa de instâncias criadas no Nitro System (Linux ou Windows) ou que tenham recursos acelerados? Se você precisar de recursos acelerados, só poderá usar o Linux com o Amazon EKS.

Número máximo de Pods

Como a cada Pod é atribuído seu próprio endereço IP, o número de endereços IP compatíveis com um tipo de instância é um fator na determinação do número de Pods que podem ser executados em uma instância. Para determinar manualmente com quantos Pods um tipo de instância é compatível, consulte Máximo recomendado de Pods do Amazon EKS para cada tipo de instância do Amazon EC2.

> **Note**
>
> Se estiver utilizando uma AMI do Amazon Linux 2 otimizada para Amazon EKS da versão `v20220406` ou mais recente, você poderá utilizar um novo tipo de instância sem fazer upgrade para a AMI mais recente. Para essas AMIs, a AMI calcula automaticamente o valor necessário de `max-pods` caso ele não esteja listado no arquivo `eni-max-pods.txt`. Tipos de instância atualmente em versão de demonstração podem não ter suporte pelo Amazon EKS por padrão. Valores para `max-pods` para esses tipos ainda precisam ser adicionados a `eni-max-pods.txt` na nossa AMI.

Os tipos de instância do AWS Nitro System opcionalmente oferecem suporte a bem mais endereços IP do que os tipos de instância que não são do Nitro System. Porém, nem todos os endereços IP atribuídos a uma instância estão disponíveis para Pods. Para atribuir um número significativamente maior de endereços IP às suas instâncias, você deve ter a versão `1.9.0` ou superior do complemento CNI da Amazon VPC instalado em seu cluster e configurado adequadamente. Para obter mais informações, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2. Para atribuir o maior número de endereços IP às suas instâncias, a versão `1.10.1` ou superior do complemento CNI da Amazon VPC deverá estar instalada em seu cluster e o cluster deverá ser implantado com a família `IPv6`.

Família IP

Você pode usar qualquer tipo de instância compatível quando usa a família `IPv4` para um cluster, o que permite ao cluster atribuir endereços `IPv4`privados aos Pods e aos serviços. Porém,

se você deseja utilizar a família `IPv6` para o seu cluster, deve usar tipos de instância do AWS Nitro System ou tipos de instância de bare metal. Apenas `IPv4` é compatível com instâncias do Windows. O cluster deve executar a versão `1.10.1` ou posterior do complemento CNI da Amazon VPC. Para obter mais informações sobre o uso de `IPv6`, consulte Endereços `IPv6` para clusters, Pods e services.

Versão do complemento Amazon VPC CNI que você está executando

A versão mais recente do plug-in Amazon VPC CNI para Kubernetes é compatível com estes tipos de instância. Talvez seja necessário atualizar a versão do complemento Amazon VPC CNI para poder aproveitar os tipos de instância mais recentes com suporte. Para obter mais informações, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS. A versão mais recente suporta os recursos mais recentes para serem usados com o Amazon EKS. As versões anteriores não suportam todos os recursos. Você pode visualizar os recursos compatíveis com as diferentes versões no Changelog no GitHub.

A Região da AWS em que você está criando os nós

Nem todos os tipos de instâncias estão disponíveis em todas as Regiões da AWS.

Se você estiver usando grupos de segurança para Pods

Se você estiver usando grupos de segurança para Pods: apenas determinados tipos de instância serão compatíveis. Para obter mais informações, consulte Grupos de segurança do Pods.

Máximo recomendado de Pods do Amazon EKS para cada tipo de instância do Amazon EC2

Como a cada Pod é atribuído seu próprio endereço IP, o número de endereços IP compatíveis com um tipo de instância é um fator na determinação do número de Pods que podem ser executados em uma instância. O Amazon EKS fornece um script que você pode baixar e executar para determinar o número máximo recomendado de Pods do Amazon EKS a serem executados em cada tipo de instância. O script usa os atributos de hardware de cada instância e opções de configuração para determinar o número máximo de Pods. Você pode usar o número retornado nessas etapas para habilitar recursos como atribuição de endereços IP a Pods de uma sub-rede diferente da sub-rede da instância e aumento significativo do número de endereços IP da instância. Se você estiver usando um grupo de nós gerenciados com vários tipos de instâncias, use um valor que funcione para todos os tipos de instâncias.

1. Baixe um script que você pode usar para calcular o número máximo de Pods para cada tipo de instância.

```
curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/templates/al2/runtime/max-pods-calculator.sh
```

2. Marque o script como executável no computador.

```
chmod +x max-pods-calculator.sh
```

3. Execute o script, substituindo *m5.large* pelo tipo de instância que você planeja implantar e *1.9.0-eksbuild.1* pela versão do complemento Amazon VPC CNI. Para determinar a versão do complemento, consulte os procedimentos de atualização em Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

```
./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
```

Veja um exemplo de saída abaixo.

```
29
```

Você pode adicionar as opções a seguir ao script para ver o máximo de Pods compatíveis ao usar recursos opcionais.

- `--cni-custom-networking-enabled`: use essa opção quando quiser atribuir endereços IP de uma sub-rede diferente da sub-rede da sua instância. Para obter mais informações, consulte Rede personalizada para pods. Adicionar essa opção ao script anterior com os mesmos valores do exemplo gera `20`.
- `--cni-prefix-delegation-enabled`: use esta opção quando quiser atribuir significativamente mais endereços IP a cada interface de rede elástica. Esse recurso requer uma instância do Amazon Linux executada no Sistema Nitro e na versão `1.9.0` ou superior do complemento CNI da Amazon VPC. Para obter mais informações, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2. Adicionar essa opção ao script anterior com os mesmos valores do exemplo gera `110`.

Você também pode executar o script com a opção `--help` para ver todas as opções disponíveis.

> **Note**
>
> O script da calculadora de Pods máximos limita o valor de retorno para `110` com base nos limites de escalabilidade do Kubernetes e nas configurações recomendadas. Se o seu tipo de instância tiver mais de 30 vCPUs, esse limite aumentará para `250`, um número baseado em testes internos da equipe de escalabilidade do Amazon EKS. Para obter mais informações, consulte a publicação no blog Plug-in do Amazon VPC CNI aumenta os limites de pods por nó.

Crie nós com imagens otimizadas predefinidas

Você pode implantar os nós com as Imagens de máquina da Amazon (AMIs) pré-criadas e otimizadas para Amazon EKS ou com suas próprias AMIs personalizadas. Para obter informações sobre cada tipo de AMI otimizada para o Amazon EKS, consulte um dos tópicos a seguir. Para instruções sobre como criar a sua própria AMI personalizada, consulte Criar uma AMI do Amazon Linux personalizada com um script.

Tópicos

- Migrar de dockershim para containerd
- Criar nós com AMIs do Amazon Linux otimizadas
- Criar nós com AMIs do Bottlerocket otimizadas
- Criar nós com AMIs do Ubuntu Linux otimizadas
- Criar nós com AMIs do Windows otimizadas

Migrar de **`dockershim`** para **`containerd`**

O Kubernetes não oferece mais suporte a `dockershim`. A equipe do Kubernetes removeu o runtime do Kubernetes versão `1.24`. Para obter mais informações, consulte O Kubernetes está migrando do Dockershim: compromissos e próximas etapas no Blog do Kubernetes.

O Amazon EKS também deixou de ser compatível com o `dockershim` a partir do Kubernetes versão `1.24`. O único runtime das AMIs do Amazon EKS oficialmente publicadas a partir da versão `1.24` é o `containerd`. Esse tópico aborda alguns detalhes, mas há mais informações disponíveis em Tudo o que você precisa saber sobre a mudança para o containerd no Amazon EKS.

Existe um plug-in `kubectl` que você pode usar para ver quais workloads do Kubernetes montam o volume de soquete do Docker. Para obter mais informações, consulte Detector para soquete do Docker Socket (DDS) no GitHub. As AMIs do Amazon EKS que executam versões do Kubernetes anteriores à `1.24` usam o Docker como runtime padrão. Porém, essas AMIS do Amazon EKS têm uma opção de sinalizador de bootstrap que você pode usar para testar as workloads em qualquer cluster compatível que use o `containerd`. Para obter mais informações, consulte Teste a migração do Amazon Linux 2 de Docker para `containerd`.

Continuaremos a publicar AMIs para as versões do Kubernetes até o final de sua data de suporte. Para obter mais informações, consulte Calendário de lançamento do Amazon EKS Kubernetes. Se você precisar de mais tempo para testar as workloads no `containerd`, use uma versão compatível anterior à `1.24`. Porém, quando quiser atualizar as AMIs oficiais do Amazon EKS para a versão `1.24` ou posteriores, certifique-se de validar que as workloads podem ser executadas no `containerd`.

O runtime `containerd` oferece performance e segurança mais confiáveis. O `containerd` é o runtime que está sendo padronizado em todo o Amazon EKS. O Fargate e o Bottlerocket já usam apenas o `containerd`. O `containerd` ajuda a minimizar o número de versões de AMI do Amazon EKS necessárias para resolver as vulnerabilidades e exposições comuns (CVEs) do `dockershim`. Como o `dockershim` já usa o `containerd` internamente, talvez não seja necessário fazer nenhuma alteração. No entanto, há algumas situações em que pode ser necessário fazer alterações:

- É necessário fazer alterações nas aplicações que montam o soquete Docker. Por exemplo, imagens de contêiner que são criadas com um contêiner são afetadas. Muitas ferramentas de monitoramento também montam o soquete do Docker. Pode ser necessário aguardar atualizações ou reimplantar workloads para monitoramento de runtime.
- Também pode ser necessário fazer alterações nas aplicações que dependem de configurações específicas do Docker. Por exemplo, o protocolo `HTTPS_PROXY` não é mais compatível. Será necessário atualizar as aplicações que usam esse protocolo. Para obter mais informações, consulte `dockerd` no Docker Docs.
- Se você usar o auxiliar de credenciais do Amazon ECR para extrair imagens, será necessário alternar para o provedor de credenciais de imagem do `kubelet`. Para obter mais informações, consulte Configure a `kubelet` image credential provider (Configurar um provedor de credenciais de imagens do) na documentação do Kubernetes.
- Como o Amazon EKS `1.24` não é mais compatível com o Docker, alguns sinalizadores que o script de bootstrap do Amazon EKS aceitava anteriormente não são mais compatíveis. Antes

de mudar para o Amazon EKS `1.24` ou posterior, você deve remover qualquer referência aos sinalizadores que não são mais compatíveis:

- `--container-runtime dockerd` (O único valor compatível é `containerd`)
- `--enable-docker-bridge`
- `--docker-config-json`

- Se você já Fluentd configurou para Container Insights, deve migrar Fluentd para Fluent Bit antes de mudar para `containerd`. Os analisadores Fluentd são configurados para analisar apenas mensagens de log no formato JSON. Ao contrário de `dockerd`, o runtime do `containerd` contêiner tem mensagens de log que não estão no formato JSON. Se você não migrar para Fluent Bit, alguns dos Fluentd's analisadores configurados gerarão muitos erros dentro do Fluentd contêiner. Para obter mais informações sobre migração, consulte Configurar Fluent Bit como um DaemonSet para enviar logs para o CloudWatch Logs.
- Se você usa uma AMI personalizada e está fazendo o upgrade para o Amazon EKS `1.24`, deve garantir que o encaminhamento de IP esteja habilitado para seus nós de processamento. Essa configuração não era necessária com o Docker, mas é necessária para o `containerd`. Ela é necessária para solucionar problemas de conectividade de rede Pod para Pod, Pod para externa ou Pod para apiserver.

  Para verificar essa configuração em um nó de processamento, execute um dos seguintes comandos:

  - `sysctl net.ipv4.ip_forward`
  - `cat /proc/sys/net/ipv4/ip_forward`

  Se a saída for `0`, execute um dos seguintes comandos para ativar a variável `net.ipv4.ip_forward` do kernel:

  - `sysctl -w net.ipv4.ip_forward=1`
  - `echo 1 > /proc/sys/net/ipv4/ip_forward`

  Para a ativação da configuração nas AMIs do Amazon EKS no runtime `containerd`, consulte `install-worker.sh` no GitHub.

Teste a migração do Amazon Linux 2 de Docker para **`containerd`**

Para a versão `1.23` do Kubernetes, é possível usar um sinalizador de inicialização opcional para habilitar o runtime do `containerd` para AMIs do AL2 otimizadas para o Amazon EKS. Esse

recurso fornece um caminho claro de migração para o `containerd` ao atualizar para a versão `1.24` ou posterior. O Amazon EKS deixou de ser compatível com o Docker a partir da versão `1.24` do Kubernetes. O runtime do `containerd` tem sido amplamente adotado na comunidade do Kubernetes e é um projeto graduado com a CNCF. Você pode testá-lo adicionando um grupo de nós a um cluster novo ou existente.

Você pode habilitar o sinalizador de boostrap criando um dos tipos de grupos de nós a seguir.

Autogerenciado

Implante o grupo de nós usando as instruções em Criar nós autogerenciados do Amazon Linux. Especifique uma AMI otimizada para o Amazon EKS e o texto a seguir para o parâmetro `BootstrapArguments`.

```
--container-runtime containerd
```

Gerenciados

Se você usar o `eksctl`, crie um arquivo denominado *`my-nodegroup`*`.yaml` com o conteúdo a seguir. Substitua *`example value`* por seus próprios valores. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Para recuperar um ID de AMI otimizada para `ami-`*`1234567890abcdef0`*, consulte Recuperar IDs de AMI do Amazon Linux recomendadas.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
  version: 1.23
managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster --container-runtime containerd
```

> **Note**
>
> Se você iniciar muitos nós ao mesmo tempo, também poderá especificar valores para os argumentos de bootstrap `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-`

`cluster-ip` para evitar erros. Para obter mais informações, consulte Especificar uma AMI.

Execute os comandos a seguir para criar um grupo de nós.

```
eksctl create nodegroup -f my-nodegroup.yaml
```

Se preferir usar uma ferramenta diferente para criar o grupo de nós gerenciados, é necessário implantar o grupo de nós usando um modelo de execução. No seu modelo de execução, especifique um ID de AMI otimizada para Amazon EKS e implante o grupo de nós usando um modelo de execução e forneça os seguintes dados de usuário. Esses dados do usuário transmitem argumentos para o arquivo `bootstrap.sh`. Para obter mais informações sobre o arquivo de bootstrap, consulte bootstrap.sh no GitHub.

```
/etc/eks/bootstrap.sh my-cluster --container-runtime containerd
```

Criar nós com AMIs do Amazon Linux otimizadas

A AMI do Amazon Linux otimizada para o Amazon EKS é desenvolvida com base no Amazon Linux 2 (AL2) e no Amazon Linux 2023 (AL2023). Está configurada para servir como imagem base para nós do Amazon EKS. A AMI é configurada para funcionar com o Amazon EKS e inclui os seguintes componentes:

- `kubelet`
- AWS IAM Authenticator
- Docker(Amazon EKS versão `1.23` e anteriores)
- `containerd`

> **Note**
>
> - É possível acompanhar eventos de segurança ou de privacidade para o AL2 no Amazon Linux Security Center ou ao se tornar assinante do Feed RSS associado. Os eventos de segurança e privacidade incluem uma visão geral do problema, quais pacotes são afetadas e como atualizar suas instâncias para corrigir o problema.

- Antes de implantar uma AMI acelerada ou do Arm, analise as informações em AMIs do Amazon Linux aceleradas e otimizadas para Amazon EKS e AMIs Amazon Linux Arm otimizadas para Amazon EKS.
- Para o Kubernetes versão1.23, você pode usar um sinalizador de bootstrap opcional para testar a migração de Docker para containerd. Para obter mais informações, consulte Teste a migração do Amazon Linux 2 de Docker para containerd.
- A partir do Kubernetes versão 1.25, não será mais possível usar instâncias P2 do Amazon EC2 com as AMIs aceleradas do Amazon Linux otimizadas para o Amazon EKS prontas para uso. Essas AMIs para o Kubernetes versões 1.25 ou posteriores serão compatíveis com drivers série NVIDIA 525 ou posterior, que são incompatíveis com as instâncias P2. No entanto, os drivers da série NVIDIA 525 ou posteriores são compatíveis com as instâncias P3, P4 e P5. Portanto, é possível usar essas instâncias com as AMIs para o Kubernetes versão 1.25 ou posterior. Antes que seus clusters do Amazon EKS sejam atualizados para a versão 1.25, migre qualquer instância P2 para instâncias P3, P4 e P5. Você também deverá atualizar proativamente suas aplicações para funcionar com a série NVIDIA 525 ou posterior. Planejamos transferir os drivers da série NVIDIA 525 ou mais recente ou para as versões 1.23 e 1.24 do Kubernetes no final de janeiro de 2024.
- Todos os grupos de nós gerenciados recém-criados em clusters na versão 1.30 ou mais recente terão como padrão automático o uso do AL2023 como sistema operacional do nó. Anteriormente, os novos grupos de nós seriam padronizados para usar o AL2. É possível continuar a usar AL2 ao escolhê-lo como o tipo de AMI durante a criação de um novo grupo de nós.
- O suporte para o AL2 será encerrado em 30 de junho de 2025. Para obter mais informações, consulte Perguntas frequentes do Amazon Linux 2.

AMIs do Amazon Linux aceleradas e otimizadas para Amazon EKS

> **Note**
>
> As AMIs aceleradas do Amazon EKS baseadas no AL2023 estarão disponíveis posteriormente. Caso tenha workloads aceleradas, você deverá continuar a usar a AMI acelerada do AL2 ou o Bottlerocket.

A AMI do Amazon Linux otimizada e acelerada para o Amazon EKS é desenvolvida sobre a AMI do Amazon Linux otimizada para o Amazon EKS padrão. A AMI está configurada para servir como uma imagem opcional para nós do Amazon EKS com a finalidade de oferecer suporte a workloads baseadas em GPU, no Inferentia e no Trainium.

Além da configuração padrão da AMI otimizada para Amazon EKS, a AMI acelerada inclui o seguinte:

- Drivers NVIDIA
- `nvidia-container-runtime`
- Driver do AWS Neuron

Para obter uma lista dos componentes mais recentes incluídos na AMI acelerada, consulte `amazon-eks-ami` Releases no GitHub.

> **Note**
>
> - A AMI acelerada, otimizada para o Amazon EKS, é compatível apenas com tipos de instâncias baseadas em GPU e Inferentia. Especifique esses tipos de instância no template do AWS CloudFormation do nó. Ao usar a AMI acelerada otimizada para o Amazon EKS, você concorda com o Contrato de licença do usuário final (EULA) da NVIDIA.
> - A AMI acelerada, otimizada para o Amazon EKS, era mencionada anteriormente como a AMI otimizada para o Amazon EKS compatível com GPU.
> - As versões anteriores da AMI acelerada, otimizada para o Amazon EKS, tinham o repositório `nvidia-docker` instalado. O repositório não está mais incluído na versão `v20200529` da AMI para o Amazon EKS e posteriores.

Para habilitar workloads baseadas no AWS Neuron (acelerador de ML)

Para obter detalhes sobre workloads de treinamento e inferência usando o Neuron no Amazon EKS, consulte as seguintes referências:

- Contêineres - Kubernetes - Começar a usar, na Documentação do AWS Neuron
- Treinamento em amostras do EKS do AWS Neuron no GitHub
- Implantar workloads de inferência de ML com o AWSInferentia no Amazon EKS

Para habilitar workloads baseadas em GPU

O procedimento a seguir descreve como executar uma workload em uma instância baseada em GPU com a AMI acelerada otimizada para o Amazon EKS.

1. Depois que seus nós de GPU ingressarem no cluster, você deverá aplicar o Plug-in de dispositivo NVIDIA para Kubernetes como um DaemonSet em seu cluster. Substitua *vX.X.X* pela versão desejada de NVIDIA/k8s-device-plugin antes de executar o comando a seguir.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

2. Você pode verificar se os nós têm GPUs alocáveis com o seguinte comando:

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

Para implantar um Pod e testar se os nós da GPU estão configurados corretamente

1. Crie um arquivo denominado `nvidia-smi.yaml` com o seguinte conteúdo: Substitua *tag* pela tag desejada para `nvidia/cuda`. Este manifesto inicia um contêiner NVIDIA CUDA que executa o `nvidia-smi` em um nó.

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: nvidia/cuda:tag
    args:
    - "nvidia-smi"
    resources:
      limits:
        nvidia.com/gpu: 1
```

2. Aplique o manifesto com o comando a seguir.

```
kubectl apply -f nvidia-smi.yaml
```

3. Após a execução do Pod ser concluída, visualize os logs com o comando a seguir.

```
kubectl logs nvidia-smi
```

Veja um exemplo de saída abaixo.

```
Mon Aug  6 20:23:31 20XX
+-----------------------------------------------------------------------------+
| NVIDIA-SMI XXX.XX                 Driver Version: XXX.XX                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:1C.0 Off |                    0 |
| N/A   46C    P0    47W / 300W |      0MiB / 16160MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

AMIs Amazon Linux Arm otimizadas para Amazon EKS

As instâncias Arm oferecem uma economia significativa para aplicações de expansão e baseadas no Arm, como servidores Web, microsserviços conteinerizados, frotas de armazenamento em cache e datastores distribuídos. Ao adicionar nós do Arm ao cluster, revise as considerações a seguir.

Considerações

- Se o cluster tiver sido implantado antes de 17 de agosto de 2020, será necessário fazer uma atualização única dos manifestos essenciais do complemento de cluster. Isso serve para que o Kubernetes possa extrair a imagem correta para cada arquitetura de hardware em uso em seu cluster. Para obter mais informações sobre como atualizar complementos do cluster, consulte Atualizar a versão do Kubernetes de um cluster do Amazon EKS. Se você implantou o cluster a

partir de 17 de agosto de 2020, os complementos CoreDNS, `kube-proxy` e Amazon VPC CNI plugin for Kubernetes já têm capacidade para várias arquiteturas.

- As aplicações implantadas em nós do Arm devem ser compiladas para Arm.
- Se tiver DaemonSets implantados em um cluster existente ou quiser implantá-los em um novo cluster no qual também queira implantar nós do Arm, verifique se o DaemonSet pode ser executado em todas as arquiteturas de hardware do cluster.
- Você pode executar grupos de nós do Arm e grupos de nós x86 no mesmo cluster. Se o fizer, considere implantar imagens de contêiner de várias arquiteturas em um repositório de contêineres, como o Amazon Elastic Container Registry, e depois adicionar seletores de nós aos manifestos para que o Kubernetes saiba em que arquitetura de hardware um Pod pode ser implantado. Para obter mais informações, consulte Enviar uma imagem de várias arquiteturas no Guia do usuário do Amazon ECR e a publicação de blog Introducing multi-architecture container images for Amazon ECR.

Mais informações

Para obter mais informações sobre o uso de AMIs Amazon Linux otimizadas para Amazon EKS, consulte as seguintes seções:

- Para usar o Amazon Linux com grupos de nós gerenciados, consulte Simplificar o ciclo de vida dos nós com grupos de nós gerenciados.
- Para iniciar nós autogerenciados do Amazon Linux, consulte Recuperar IDs de AMI do Amazon Linux recomendadas.
- Para obter informações sobre versões, consulte Recuperar informações da versão da AMI do Amazon Linux.
- Para recuperar os IDs mais recentes das AMIs Amazon Linux otimizadas para Amazon EKS, consulte Recuperar IDs de AMI do Amazon Linux recomendadas.
- Para scripts de código aberto usados para criar a AMI otimizada do Amazon EKS, consulte Criar uma AMI do Amazon Linux personalizada com um script.

Atualização do Amazon Linux 2 para o Amazon Linux 2023

A AMI otimizada para o Amazon EKS está disponível em duas famílias baseadas no AL2 e no AL2023. O AL2023 é um novo sistema operacional baseado no Linux que foi projetado para fornecer um ambiente seguro, estável e de alta performance para as aplicações em nuvem. Ele

corresponde à próxima geração do Amazon Linux da Amazon Web Services e está disponível em todas as versões do Amazon EKS com suporte, incluindo as versões `1.23` e `1.24` que estão em suporte estendido. As AMIs aceleradas do Amazon EKS baseadas no AL2023 estarão disponíveis posteriormente. Se você tiver workloads aceleradas, deverá continuar a usar a AMI acelerada do AL2 ou o Bottlerocket.

O AL2023 oferece diversas melhorias em relação ao AL2. Para obter uma comparação completa, consulte Comparing AL2 and Amazon Linux 2023 no Guia do usuário do Amazon Linux 2023. Vários pacotes foram adicionados, atualizados e removidos em relação ao AL2. É altamente recomendável testar as aplicações com o AL2023 antes de realizar a atualização. Para obter uma lista de todas as alterações de pacote no AL2023, consulte Package changes in Amazon Linux 2023 nas Notas de lançamento do Amazon Linux 2023.

Além dessas alterações, você deve estar ciente do seguinte:

- O AL2023 introduz um novo processo de inicialização do nó `nodeadm` que usa um esquema de configuração YAML. Se estiver usando grupos de nós autogerenciados ou uma AMI com um modelo de inicialização, será necessário fornecer, de forma explícita, metadados de cluster adicionais ao criar um novo grupo de nós. Veja a seguir um exemplo dos parâmetros mínimos necessários, em que `apiServerEndpoint`, `certificateAuthority` e `cidr` do serviço passaram a ser necessários:

  ```
  ---
  apiVersion: node.eks.aws/v1alpha1
  kind: NodeConfig
  spec:
    cluster:
      name: my-cluster
      apiServerEndpoint: https://example.com
      certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
      cidr: 10.100.0.0/16
  ```

  No AL2, os metadados desses parâmetros eram revelados na chamada de API `DescribeCluster` do Amazon EKS. Com o AL2023, esse comportamento foi alterado, uma vez que a chamada de API adicional corre o risco de sofrer controle de utilização durante grandes aumentos de escala vertical para nós. Essa alteração não afetará você se estiver usando grupos de nós gerenciados sem um modelo de inicialização ou se estiver usando o Karpenter. Para obter mais informações sobre `certificateAuthority` e `cidr` do serviço, consulte `DescribeCluster` na Referência de APIs do Amazon EKS.

- O Docker não é compatível no AL2023 para todas as versões compatíveis do Amazon EKS. O suporte para o Docker foi encerrado e removido com a versão `1.24` ou com versões posteriores do Amazon EKS no AL2. Para obter mais informações sobre a descontinuação, consulte Amazon EKS ended support for Dockershim.
- A versão `1.16.2` ou as versões posteriores do plug-in CNI da Amazon VPC são necessárias para o AL2023.
- O AL2023 requer `IMDSv2` por padrão. O `IMDSv2` tem diversos benefícios que ajudam a aprimorar a postura de segurança. Ele usa um método de autenticação orientado por sessão que requer a criação de um token secreto em uma solicitação HTTP PUT simples para iniciar a sessão. O token de uma sessão pode ter validade de um segundo a seis horas. Para obter mais informações sobre como fazer a transição do `IMDSv1` para o `IMDSv2`, consulte Transition to using Instance Metadata Service Version 2 e Get the full benefits of IMDSv2 and disable IMDSv1 across your AWS infrastructure. Se desejar usar o `IMDSv1`, você ainda poderá fazê-lo ao substituir manualmente as configurações usando as propriedades de inicialização da opção de metadados da instância.

  > **Note**
  >
  > Para o `IMDSv2`, a contagem de saltos padrão para os grupos de nós gerenciados é definida como 1. Isso significa que os contêineres não terão acesso às credenciais do nó usando o IMDS. Se você precisar de acesso de contêiner às credenciais do nó, ainda poderá obtê-lo ao substituir manualmente `HttpPutResponseHopLimit` em um modelo de inicialização personalizado do Amazon EC2 e aumentá-lo para 2. Como alternativa, é possível usar a Identidade de Pod do Amazon EKS para fornecer credenciais em vez de usar o `IMDSv2`.

- O AL2023 apresenta a próxima geração de hierarquia de grupo de controle unificada (`cgroupv2`). A hierarquia `cgroupv2` é usada para implementar um runtime de contêiner e usar `systemd`. Embora o AL2023 ainda inclua códigos que podem fazer o sistema funcionar ao usar `cgroupv1`, esta não é uma configuração recomendada ou com suporte. Essa configuração será completamente removida em uma versão principal futura do Amazon Linux.
- O `eksctl` versão `0.176.0` ou superior é necessário para o `eksctl` oferecer suporte ao AL2023.

Para grupos de nós gerenciados anteriormente existentes, é possível realizar uma atualização no local ou uma atualização azul/verde, dependendo de como você está usando um modelo de inicialização:

- Caso esteja usando uma AMI personalizada com um grupo de nós gerenciados, você poderá realizar uma atualização local ao alterar o ID da AMI no modelo de inicialização. Você deve garantir que as aplicações e quaisquer dados do usuário sejam transferidos para o AL2023 antes de executar essa estratégia de atualização.
- Se você estiver usando grupos de nós gerenciados com o modelo de inicialização padrão ou com um modelo de inicialização personalizado que não especifica o ID da AMI, será necessário fazer upgrade usando uma estratégia azul/verde. Uma atualização azul/verde, normalmente, é mais complexa e envolve a criação de um grupo de nós totalmente novo no qual você especificaria o AL2023 como o tipo de AMI. O novo grupo de nós precisará ser configurado cuidadosamente para garantir que todos os dados personalizados do grupo de nós do AL2 sejam compatíveis com o novo sistema operacional. Depois que o novo grupo de nós tiver sido testado e validado com as aplicações, os Pods poderão ser migrados do grupo de nós antigo para o novo grupo de nós. Depois que a migração for concluída, você poderá excluir o grupo de nós antigo.

Se estiver usando o Karpenter e desejar usar o AL2023, você precisará modificar o campo `EC2NodeClass` `amiFamily` com AL2023. Por padrão, a Oscilação está habilitada no Karpenter. Isso significa que assim que o campo `amiFamily` for alterado, o Karpenter atualizará automaticamente seus nós de processamento para a AMI mais recente, quando disponível.

Recuperar informações da versão da AMI do Amazon Linux

O versionamento das AMIs do Amazon Linux otimizadas para Amazon EKS é feito com base na versão do Kubernetes e na data de lançamento da AMI, no seguinte formato:

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

Cada versão de AMI inclui várias versões do `kubelet`, do kernel do Linux e do `containerd`. A AMI acelerada também inclui várias versões do driver NVIDIA. Agora, essas informações de versão estão disponível no Changelog em GitHub.

Recuperar IDs de AMI do Amazon Linux recomendadas

Você pode recuperar programaticamente o ID da imagem de máquina da Amazon (AMI) para AMIs otimizadas para o Amazon EKS consultando a API Repositório de parâmetros do AWS Systems Manager. Esse parâmetro elimina a necessidade de pesquisar manualmente IDs de AMIs otimizadas para o Amazon EKS. Para obter mais informações sobre a API Systems Manager Parameter Store, consulte GetParameter. A entidade principal do IAM que você usou deve ter a permissão `ssm:GetParameter` do IAM para recuperar os metadados da AMI otimizada do Amazon EKS.

Você pode recuperar o ID de imagem da mais recente AMI do Amazon Linux recomendada otimizada para o Amazon EKS com o comando a seguir, que usa o subparâmetro `image_id`. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado:

- Substitua *kubernetes-version* por qualquer versão compatível do Amazon EKS.
- Substitua *ami-type* por uma das opções a seguir. Para obter mais informações sobre os tipos de instâncias do Amazon EC2, consulte Tipos de instância do Amazon EC2.
  - Use `amazon-linux-2023/x86_64/standard` para instâncias baseadas em x86 do Amazon Linux 2023 (AL2023).
  - Use `amazon-linux-2023/arm64/standard` para instâncias ARM do AL2023.
  - Use `amazon-linux-2` para instâncias baseadas em x86 do Amazon Linux 2 (AL2).
  - Use `amazon-linux-2-arm64` para instâncias ARM do AL2, como instâncias baseadas no AWS Graviton.
  - Use `amazon-linux-2-gpu` para instâncias aceleradas por GPU do AL2.
- Substitua *region-code* por uma Região da AWS do Amazon EKS compatível para a qual você deseja o ID da AMI.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/kubernetes-version/ami-
type/recommended/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Aqui está um exemplo de comando após as substituições do espaço reservado terem sido feitas.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.30/amazon-linux-2023/
x86_64/standard/recommended/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

Veja um exemplo de saída abaixo.

```
ami-1234567890abcdef0
```

Criar uma AMI do Amazon Linux personalizada com um script

O Amazon Elastic Kubernetes Service (Amazon EKS) tem scripts de código aberto que são usados para criar a AMI otimizada para o Amazon EKS. Esses scripts de compilação estão disponíveis no GitHub.

A AMI do Amazon Linux otimizada para o Amazon EKS é desenvolvida com base no Amazon Linux 2 (AL2) e no Amazon Linux 2023 (AL2023), especificamente para uso como um nó em clusters do Amazon EKS. É possível usar esse repositório para visualizar os detalhes de como a equipe do Amazon EKS configura o `kubelet`, o runtime e o AWS IAM Authenticator para Kubernetes e cria do zero sua própria AMI baseada no Amazon Linux.

O repositório de scripts de compilação inclui um modelo do HashiCorp packer e scripts de compilação para gerar uma AMI. Esses scripts são a fonte da dados verdadeiros para compilações de AMI otimizadas para o Amazon EKS, assim sendo, você pode acompanhar o repositório do GitHub para monitorar alterações em nossas AMIs. Por exemplo, talvez você queira que sua AMI use a mesma versão do Docker que a equipe do Amazon EKS usa para a AMI oficial.

O repositório do GitHub também contém o script de bootstrap e script de nodeadm que é executado no momento da inicialização para configurar os dados do certificado da instância, o endpoint do ambiente de gerenciamento, o nome do cluster e muito mais.

Além disso, o repositório do GitHub contém nossos modelos AWS CloudFormation de nó do Amazon EKS. Com esses modelos, fica mais fácil ativar uma instância executando a AMI otimizada do Amazon EKS e registrá-la em um cluster.

Para obter mais informações, consulte os repositórios no GitHub em https://github.com/awslabs/amazon-eks-ami.

O AL2 otimizado para o Amazon EKS contém um sinalizador de inicialização opcional para habilitar o runtime do `containerd`.

Usar transcodificação de vídeo VT1 acelerada por hardware

As AMIs personalizadas do Amazon Linux no Amazon EKS são compatíveis com a família de instâncias de transcodificação de vídeo VT1 para Amazon Linux 2 (AL2), Ubuntu 18 Ubuntu e 20. O VT1 é compatível com placas de transcodificação de mídia Xilinx U30 com codecs H.264/AVC e H.265/HEVC acelerados. Para ter o benefício dessas instâncias aceleradas, você deve seguir estas etapas:

1. Crie e inicie uma AMI de base usando o AL2, o Ubuntu 18 ou o Ubuntu 20.
2. Depois que a AMI de base for iniciada, instale o driver XRT e o runtime no nó.
3. Criar um cluster do Amazon EKS..
4. Instale o plug-in FPGA do Kubernetes no cluster.

```
kubectl apply -f fpga-device-plugin.yml
```

O plugin agora anuncia dispositivos Xilinx U30 por nó em seu cluster do Amazon EKS. Você pode usar a imagem do docker FFMPEG para executar workloads de transcodificação no cluster do Amazon EKS.

Usar workloads do Habana Deep Learning (DL1)

As AMIs personalizadas do Amazon Linux 2 (AL2) no Amazon EKS podem oferecer suporte a workloads de aprendizado profundo em grande escala por meio de configurações adicionais e complementos do Kubernetes. Este documento descreve os componentes necessários para configurar uma solução genérica do Kubernetes para uma configuração on-premises ou como linha de base em uma configuração maior na nuvem. Para suportar essa função, você terá que executar as seguintes etapas em seu ambiente personalizado:

- Drivers SynapaseAI® Software carregados no sistema: incluídos nas AMIs disponíveis no Github.
- O plug-in de dispositivo Habana: um DaemonSet que permite habilitar automaticamente o registro de dispositivos Habana em seu cluster do Kubernetes e rastrear a integridade dos dispositivos.
- Helm 3.x
- Chart do Helm para instalar o operador MPI.
- Operador MPI

1. Crie e inicie uma AMI de base usando o AL2, o Ubuntu 18 ou o Ubuntu 20.
2. Siga estas instruções para configurar o ambiente para o DL1.

Criar nós com AMIs do Bottlerocket otimizadas

O Bottlerocket é uma distribuição do Linux de código aberto patrocinada e apoiada pela AWS. O Bottlerocket foi criado especificamente para hospedar workloads de contêineres. Com o Bottlerocket, é possível melhorar a disponibilidade de implantações em contêineres e reduzir os custos operacionais automatizando as atualizações em sua infraestrutura de contêineres. O Bottlerocket inclui somente o software essencial para executar contêineres, o que melhora o uso de recursos, reduz as ameaças à segurança e diminui a sobrecarga de gerenciamento. A Bottlerocket AMI inclui `containerd`, `kubelet`, e o AWS IAM Authenticator. Além de grupos de nós gerenciados e nós autogerenciados, o Bottlerocket também é compatível no Karpenter.

Vantagens

Usar o Bottlerocket com seu cluster do Amazon EKS tem as seguintes vantagens:

- Maior disponibilidade com menor custo operacional e menor complexidade de gerenciamento: o Bottlerocket ocupa menos recursos, tem menor tempo de inicialização e é menos vulnerável a ameaças à segurança do que outras distribuições do Linux. O espaço menor ocupado pelo Bottlerocket's ajuda a reduzir custos usando menos recursos de armazenamento, computação e rede.
- Segurança aprimorada com atualizações automáticas do sistema operacional: as atualizações do Bottlerocket são aplicadas como uma única unidade que pode ser revertida, se necessário. Isso elimina o risco de atualizações corrompidas ou com falha que podem deixar o sistema em um estado inutilizável. Com o Bottlerocket, as atualizações de segurança podem ser aplicadas automaticamente assim que estiverem disponíveis de forma minimamente disruptiva e revertidas se ocorrerem falhas.
- Suporte premium: as compilações do Bottlerocket no Amazon EC2 fornecidas pela AWS são cobertas pelos mesmos planos de suporte da AWS Support que também cobrem os serviços da AWS, como o Amazon EC2, o Amazon EKS e o Amazon ECR.

Considerações

Considere o seguinte ao usar o Bottlerocket para seu tipo de AMI:

- Bottlerocket oferece suporte a instâncias do Amazon EC2 com processadores `x86_64` e `arm64`. O AMI Bottlerocket não é recomendado para uso com instâncias do Amazon EC2 com um chip Inferentia.
- As imagens do Bottlerocket não incluem um servidor ou shell SSH. Você pode usar métodos de acesso fora de banda para permitir o SSH. Essas abordagens habilitam o contêiner do administrador e a aprovação de algumas etapas de configuração de bootstrapping com dados de usuário. Para obter mais informações, consulte as seguintes seções no tópico Bottlerocket OS no GitHub:
  - Exploration (Exploração)
  - Admin container (Contêiner Admin)
  - Configurações do Kubernetes
- O Bottlerocket usa diferentes tipos de contêineres:

- Por padrão, um contêiner de controle está habilitado. Esse contêiner executa o agente do AWS Systems Manager que você pode usar para executar comandos ou iniciar sessões de shell em instâncias Bottlerocket do Amazon EC2. Para obter mais informações, consulte Configurar gerenciador de sessões no Guia do usuário do AWS Systems Manager.
- Se uma chave SSH foi fornecida ao criar o grupo de nós, um contêiner de administrador está habilitado. Recomendamos usar o contêiner administrador somente para casos de desenvolvimento e teste. Não recomendamos usá-lo em ambientes de produção. Para obter mais informações, consulte Admin container(Contêiner administrador) no GitHub.

Mais informações

Para obter mais informações sobre o uso de AMIs Bottlerocket otimizadas para Amazon EKS, consulte as seguintes seções:

- Para obter mais detalhes sobre o Bottlerocket, consulte a documentação do Bottlerocket.
- Para obter recursos de informações da versão, consulte Recuperar informações da versão da AMI do Bottlerocket.
- Para usar o Bottlerocket com grupos de nós gerenciados, consulte Simplificar o ciclo de vida dos nós com grupos de nós gerenciados.
- Para iniciar nós Bottlerocket autogerenciados, consulte Criar nós Bottlerocket autogerenciados.
- Para recuperar os IDs mais recentes das AMIs Bottlerocket otimizadas para Amazon EKS, consulte Recuperar IDs de AMI do Bottlerocket recomendadas.
- Para obter detalhes sobre suporte para conformidade, consulte Atender a requisitos de conformidade com o Bottlerocket.

Recuperar informações da versão da AMI do Bottlerocket

Cada versão de AMI do Bottlerocket inclui várias versões de `kubelet`, the Bottlerocket kernel e `containerd`. As variantes de AMIs aceleradas também incluem várias versões do driver NVIDIA. Você pode encontrar essas informações sobre versões no tópico OS da documentação do Bottlerocket. Nessa página, navegue até o subtópico de informações da versão aplicável.

Às vezes, a documentação do Bottlerocket pode ficar aquém das versões disponíveis no GitHub. Você pode encontrar uma lista de alterações das versões mais recentes nas versões no GitHub.

Recuperar IDs de AMI do Bottlerocket recomendadas

Você pode recuperar o ID da imagem de máquina da Amazon (AMI) para AMIs otimizadas do Amazon EKS consultando a API Parameter Store do AWS Systems Manager. Ao usar esse parâmetro, não será necessário pesquisar manualmente IDs de AMIs otimizadas para o Amazon EKS. Para obter mais informações sobre a API Systems Manager Parameter Store, consulte GetParameter. A entidade principal do IAM que você usou deve ter a permissão `ssm:GetParameter` do IAM para recuperar os metadados da AMI otimizada do Amazon EKS.

Você pode recuperar o ID de imagem da mais recente AMI do Bottlerocket otimizada recomendada para o Amazon EKS com o comando da AWS CLI a seguir, que usa o subparâmetro `image_id`. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado:

- Substitua *`kubernetes-version`* por qualquer versão compatível do Amazon EKS.
- Substitua *`-flavor`* por uma das opções a seguir.
  - Remova *`-flavor`* para variantes sem uma GPU.
  - Use `-nvidia` para variantes habilitadas para GPU.
- Substitua *`architecture`* por uma das opções a seguir.
  - Use `x86_64` para instâncias baseadas em `x86`.
  - Use `arm64` para instâncias ARM.
- Substitua *`region-code`* por uma Região da AWS do Amazon EKS compatível para a qual você deseja o ID da AMI.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-kubernetes-version-
flavor/architecture/latest/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Aqui está um exemplo de comando após as substituições do espaço reservado terem sido feitas.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.30/x86_64/latest/
image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

Veja um exemplo de saída abaixo.

```
ami-1234567890abcdef0
```

Atender a requisitos de conformidade com o Bottlerocket

O Bottlerocket está em conformidade com as recomendações definidas por várias organizações:

- Existe um benchmark CIS definido para o Bottlerocket. Em uma configuração padrão, a imagem do Bottlerocket tem a maioria dos controles exigidos pelo perfil de configuração CIS Nível 1. É possível implementar os controles necessários para um perfil de configuração CIS de nível 2. Para obter mais informações, consulte Validar a AMI Bottlerocket otimizada para o Amazon EKS com base no benchmark CIS, no blog da AWS.
- O conjunto de recursos otimizado e a superfície de ataque reduzida significam que as instâncias do Bottlerocket exigem menos configuração para atender aos requisitos do PCI DSS. O CIS Benchmark for Bottlerocket é um excelente recurso para orientação de fortalecimento e é compatível com requisitos de padrões de configuração segura de acordo com o requisito 2.2 do PCI DSS. Também é possível usar o Fluent Bit para atender aos seus requisitos de registro de auditoria em nível de sistema operacional de acordo com o requisito 10.2 do PCI DSS. O AWS publica instâncias novas (corrigidas) do Bottlerocket periodicamente para ajudar a atender aos requisitos 6.2 do PCI DSS (para v3.2.1) e 6.3.3 (para v4.0).
- O Bottlerocket é um recurso qualificado para HIPAA autorizado para uso com workloads regulamentadas tanto para o Amazon EC2 quanto para o Amazon EKS. Para obter mais informações, consulte o whitepaper Architecting for HIPAA Security and Compliance on Amazon EKS (Arquitetura para segurança e conformidade HIPAA no Amazon EKS).

Criar nós com AMIs do Ubuntu Linux otimizadas

A Canonical fez uma parceria com o Amazon EKS para criar AMIs de nós que você pode usar nos clusters.

A Canonical fornece uma imagem de sistema operacional de nó do Kubernetes para fins específicos. A imagem minimizada do Ubuntu é otimizada para o Amazon EKS e inclui o kernel do AWS personalizado que é desenvolvido em conjunto com a AWS. Para obter mais informações, consulte Ubuntu no Amazon Elastic Kubernetes Service (EKS) e Criar nós Ubuntu Linux autogerenciados. Para obter informações sobre suporte, consulte a seção Software de terceiros nas Perguntas frequentes sobre o Suporte Premium da AWS.

Criar nós com AMIs do Windows otimizadas

As AMIs do Windows otimizadas para o Amazon EKS são desenvolvidas com base no Windows Server 2019 e no Windows Server 2022. Elas são configuradas para servirem como imagem base para nós do Amazon EKS. As AMIs incluem por padrão os seguintes componentes:

- `kubelet`
- `kube-proxy`
- Autenticador do AWS IAM para o Kubernetes
- `csi-proxy`
- `containerd`

> **Note**
>
> É possível rastrear eventos de segurança ou privacidade do Windows Server com o Guia de atualização de segurança da Microsoft.

O Amazon EKS oferece AMIs que são otimizadas para contêineres do Windows nas seguintes variantes:

- AMI do Windows Server 2019 Core otimizada para o Amazon EKS
- AMI completa do Windows Server 2019 Core otimizada para Amazon EKS
- AMI do Windows Server 2022 Core otimizada para o Amazon EKS
- AMI do Windows Server 2022 Full otimizada para o Amazon EKS

> **Important**
>
> - A AMI Windows Server 20H2 Core otimizada para Amazon EKS foi descontinuada. Nenhuma nova versão dessa AMI será lançada.
> - Por padrão, para garantir que você tenha as atualizações de segurança mais recentes, o Amazon EKS mantém AMIs do Windows otimizadas pelos últimos quatro meses. Cada nova AMI estará disponível por quatro meses a partir do lançamento inicial. Após esse período, as AMIs mais antigas se tornarão privadas e não estarão mais acessíveis.

Incentivamos o uso das AMIs mais recentes para evitar vulnerabilidades de segurança e a perda de acesso às AMIs mais antigas que atingiram o fim da vida útil com suporte. Embora não possamos garantir que poderemos fornecer acesso a AMIs que se tornaram privadas, você pode solicitar acesso ao preencher um tíquete no AWS Support.

Calendário de lançamento

A tabela a seguir lista as datas de lançamento e término da compatibilidade para versões do Windows no Amazon EKS. Se uma data de término estiver em branco, é porque a versão ainda tem suporte.

Versão do Windows	Lançamento do Amazon EKS	Fim do suporte para o Amazon EKS
Windows Server 2022 Core	10/17/2022	
Windows Server 2022 Full	10/17/2022	
Windows Server 20H2 Core	8/12/2021	8/9/2022
Windows Server 2004 Core	8/19/2020	12/14/2021
Windows Server 2019 Core	10/7/2019	
Windows Server 2019 Full	10/7/2019	
Windows Server 1909 Core	10/7/2019	12/8/2020

Parâmetros de configuração do script de bootstrap

Quando você cria um nó do Windows, existe um script nesse nó que permite configurar parâmetros diferentes. Dependendo da configuração, esse script pode ser encontrado no nó em um local semelhante a: `C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1`. Você pode especificar valores de parâmetros personalizados definindo-os como argumentos para o script de bootstrap. Por exemplo, você pode atualizar os dados do usuário no modelo de execução. Para obter mais informações, consulte Dados do usuário do Amazon EC2.

O script inclui os seguintes parâmetros de linha de comando:

- `-EKSClusterName` – Especifica o nome do cluster do Amazon EKS no qual esse nó de processamento deve ingressar.
- `-KubeletExtraArgs` – Especifica argumentos extras para `kubelet` (opcional).
- `-KubeProxyExtraArgs` – Especifica argumentos extras para `kube-proxy` (opcional).
- `-APIServerEndpoint` – Especifica o endpoint do servidor de API do cluster do Amazon EKS (opcional). Válido somente quando usado com `-Base64ClusterCA`. Ignorar chamadas `Get-EKSCluster`.
- `-Base64ClusterCA` – Especifica o conteúdo da CA do cluster codificado em base64 (opcional). Válido somente quando usado com `-APIServerEndpoint`. Ignorar chamadas `Get-EKSCluster`.
- `-DNSClusterIP` – Substitui o endereço IP a ser usado para consultas DNS dentro do cluster (opcional). O padrão é `10.100.0.10` ou `172.20.0.10` com base no endereço IP da interface primária.
- `-ServiceCIDR`: substitui o intervalo de endereços IP do serviço Kubernetes com base no qual os serviços de cluster são endereçados. O padrão é `172.20.0.0/16` ou `10.100.0.0/16` com base no endereço IP da interface primária.
- `-ExcludedSnatCIDRs`: uma lista de CIDRs `IPv4` a serem excluídos da Source Network Address Translation (SNAT). Isso significa que o IP privado do pod que é endereçável pela VPC não seria convertido no endereço IP do endereço `IPv4` primário da ENI da instância para o tráfego de saída. Por padrão, o CIDR `IPv4` da VPC para o nó Windows do Amazon EKS é adicionado. Especificar CIDRs para esse parâmetro também exclui adicionalmente os CIDRs especificados. Para obter mais informações, consulte SNAT para Pods.

Além dos parâmetros de linha de comando, você também pode especificar alguns parâmetros de variável de ambiente. Ao especificar um parâmetro de linha de comando, ele tem precedência sobre a respectiva variável de ambiente. A(s) variável(is) de ambiente deve(m) ser definida(s) como escopo de máquina (ou sistema), já que o script de inicialização somente lerá variáveis de escopo de máquina.

O script leva em consideração as seguintes variáveis de ambiente:

- `SERVICE_IPV4_CIDR`— Consulte o parâmetro da linha de comando `ServiceCIDR` para obter a definição.
- `EXCLUDED_SNAT_CIDRS`— Deve ser uma strings separada por vírgula. Consulte o parâmetro da linha de comando `ExcludedSnatCIDRs` para obter a definição.

Compatibilidade com autenticação do gMSA

Os Pods do Amazon EKS Windows permitem diferentes tipos de autenticação de Conta de serviço gerenciada em grupo (gMSA).

- O Amazon EKS oferece suporte a identidades de domínio do Active Directory para autenticação. Para obter mais informações sobre a gMSA vinculada ao domínio, consulte Windows Autenticação em podsWindows do Amazon EKS, no blog da AWS.
- O Amazon EKS oferece um plug-in que permite que nós Windows não vinculados ao domínio recuperem credenciais gMSA com uma identidade de usuário portátil. Para obter mais informações sobre a gMSA sem domínio, consulte Autenticação Windows sem domínio em podsWindows do Amazon EKS, no blog da AWS.

Imagens de contêiner em cache

As AMIs do Windows otimizadas para Amazon EKS têm determinadas imagens de contêiner armazenadas em cache para o runtime `containerd`. Imagens de contêiner são armazenadas em cache ao criar AMIs personalizadas com o uso componentes de compilação gerenciados pela Amazon. Para obter mais informações, consulte Usar o componente de compilação gerenciado pela Amazon.

As seguintes imagens de contêiner em cache são para o runtime `containerd`:

- `amazonaws.com/eks/pause-windows`
- `mcr.microsoft.com/windows/nanoserver`
- `mcr.microsoft.com/windows/servercore`

Mais informações

Para obter mais informações sobre o uso de AMIs Windows otimizadas para Amazon EKS, consulte as seguintes seções:

- Para usar o Windows com grupos de nós gerenciados, consulte Simplificar o ciclo de vida dos nós com grupos de nós gerenciados.
- Para iniciar nós Windows autogerenciados, consulte Criar nós Microsoft Windows autogerenciados.

- Para obter informações sobre versões, consulte Recuperar informações da versão da AMI do Windows.
- Para recuperar os IDs mais recentes das AMIs Windows otimizadas para Amazon EKS, consulte Recuperar IDs de AMI do Microsoft Windows recomendadas.
- Para usar o Amazon EC2 Image Builder para criar AMIs Windows personalizadas otimizadas para Amazon EKS, consulte Crie uma AMI do Windows personalizada com o Image Builder.
- Para conhecer as práticas recomendadas, consulte Amazon EKS optimized Windows AMI management no Guia das práticas recomendads do EKS.

Criar nós autogerenciados do Windows Server 2022 com o **`eksctl`**

Você pode usar o **`test-windows-2022.`**`yaml` a seguir como referência para criar nós autogerenciados do Windows Server 2022. Substitua *`example value`* por seus próprios valores.

> **Note**
>
> Você deve usar um `eksctl` versão `0.116.0` ou posterior para executar nós autogerenciados do Windows Server 2022.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-2022-cluster
  region: region-code
  version: '1.30'

nodeGroups:
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2022FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
  - name: linux-ng
    amiFamily: AmazonLinux2
    minSize: 2
    maxSize: 3
```

Os grupos de nós podem ser criados com o uso do seguinte comando:

```
eksctl create cluster -f test-windows-2022.yaml
```

Recuperar informações da versão da AMI do Windows

> **Important**
>
> O Extended Support para AMIs do Windows otimizadas do Amazon EKS que são publicadas pela AWS não está disponível para o Kubernetes versão `1.23`, mas está disponível para o Kubernetes versão `1.24` e superior.

Este tópico lista as versões das AMIs do Windows otimizadas para o Amazon EKS e as versões correspondentes do `kubelet`, `containerd` e `csi-proxy`.

Os metadados da AMI otimizada para o Amazon EKS, incluindo o ID da AMI, podem ser recuperados de maneira programática em cada variante. Para obter mais informações, consulte Recuperar IDs de AMI do Microsoft Windows recomendadas.

As versões das AMIs são definidas pela versão do Kubernetes e pela data de lançamento da AMI, no seguinte formato:

```
k8s_major_version.k8s_minor_version-release_date
```

> **Note**
>
> Os grupos de nós gerenciados do Amazon EKS são compatíveis com as versões das AMIs do Windows de novembro de 2022 e posteriores.

AMI do Windows Server 2022 Core otimizada para o Amazon EKS

As tabelas a seguir listam as versões atuais e anteriores da AMI do Windows Server 2022 Core otimizada para o Amazon EKS.

Kubernetes version 1.30

Kubernetes versão **1.30**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Inclui patches para o CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Atualizado containerd para 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes versão **1.29**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Atualizado containerd para 1.7.11. Atualizado kubelet para 1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Reconstrução do

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
				CNI e do `csi-proxy` usando o `golang 1.22.1`.
`1.29-2024.03.12`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.13`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.06`	`1.29.0`	`1.6.25`	`1.1.2`	Corrigido um erro em que a imagem de pausa era excluída incorretamente pelo processo de coleta de resíduos do `kubelet`.
`1.29-2024.01.11`	`1.29.0`	`1.6.18`	`1.1.2`	Atualização autônoma do Windows KB5034439 excluída nas AMIs principais do Windows Server 2022. O KB se aplica somente às instalações do Windows com uma partição específica do WinRE, que não estão incluídas em nenhuma das nossas AMIs otimizadas do Windows para Amazon EKS.

Kubernetes version 1.28

Kubernetes versão **1.28**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.28.8.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.11	1.28.5	1.6.18	1.1.2	Atualização autônoma do Windows KB5034439 excluída nas AMIs principais do Windows Server 2022. O KB se aplica somente às instalações do Windows com uma partição específica do WinRE, que não estão incluídas em nenhuma

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
				das nossas AMIs otimizadas do Windows para Amazon EKS.
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes versão **1.27**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2024.07.10	1.27.15	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.27.12.
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.11	1.27.9	1.6.18	1.1.2	Atualização autônoma do Windows KB5034439 excluída nas AMIs principais do Windows Server 2022. O KB se aplica somente às instalações do Windows com uma partição específica do WinRE, que não estão incluídas em nenhuma das nossas AMIs otimizadas do Windows para Amazon EKS.
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	Inclui patches para o CVE-2023-3676 , CVE-2023-3893 e CVE-2023-3955 .
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	Foi corrigido o containers-roadmap problema #2042, que fazia com que os nós falhassem ao extrair imagens privadas do Amazon ECR.
1.27-2023.05.17	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes versão **1.26**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2024.07.10	1.26.15	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.26-2024.06.17	1.26.15	1.7.11	1.1.2	Atualizado containerd para 1.7.11.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.26.15.
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.26-2024.03.12	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.11	1.26.12	1.6.18	1.1.2	Atualização autônoma do Windows KB5034439 excluída nas AMIs principais do Windows Server 2022. O KB se aplica somente às instalações do Windows com uma partição específica do WinRE, que não estão incluídas em nenhuma das nossas AMIs otimizadas do Windows para Amazon EKS.
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	Atualizado `containerd` para `1.6.18`. Atualizado `kubelet` para `1.26.9`. Foram adicionadas novas variáveis do ambiente de script de bootstrap (`SERVICE_IPV4_CIDR` e `EXCLUDED_SNAT_CIDRS`).
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Inclui patches para o `CVE-2023-3676`, `CVE-2023-3893` e `CVE-2023-3955`.
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Atualizado Kubernetes para 1.26.4. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes versão **1.25**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.25-2024.07.10	1.25.16	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.25-2024.06.17	1.25.16	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	Atualizado containerd para 1.6.28.
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.25-2024.03.12	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.11	1.25.16	1.6.18	1.1.2	Atualização autônoma do Windows KB5034439 excluída nas AMIs principais do Windows Server 2022. O KB se aplica somente às instalações do Windows com uma partição específica do WinRE, que não estão incluídas em nenhuma das nossas AMIs otimizadas do Windows para Amazon EKS.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Atualizado kubelet para 1.25.14. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	Inclui patches para o CVE-2023-3676 , CVE-2023-3893 e CVE-2023-3955 .
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	Atualizado Kubernetes para 1.25.9. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Instalação de um plug-in gMSA sem domínio para facilitar a autenticação gMSA para contêineres do Windows no Amazon EKS.
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes versão **1.24**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2024.07.10	1.24.17	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.24-2024.06.17	1.24.17	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	Atualizado containerd para 1.6.28.
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.24-2024.03.12	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.11	1.24.17	1.6.18	1.1.2	Atualização autônoma do Windows KB5034439 excluída

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
				nas AMIs principais do Windows Server 2022. O KB se aplica somente às instalações do Windows com uma partição específica do WinRE, que não estão incluídas em nenhuma das nossas AMIs otimizadas do Windows para Amazon EKS.
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Atualizado kubelet para 1.24.17. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	Inclui patches para o CVE-2023-3676 , CVE-2023-3893 e CVE-2023-3955 .

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.20	1.24.13	1.6.6	1.1.1	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	Atualizado Kubernetes para 1.24.13. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Instalação de um plug-in gMSA sem domínio para facilitar a autenticação gMSA para contêineres do Windows no Amazon EKS.
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	A versão do Kubernetes sofreu um downgrad para a versão 1.24.7 porque a 1.24.10 tem um problema relatado no kube-proxy .
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.13	1.24.7	1.6.6	1.1.1	
1.24-2022.10.11	1.24.7	1.6.6	1.1.1	

AMI do Windows Server 2022 Full otimizada para o Amazon EKS

As tabelas a seguir listam as versões atuais e anteriores da AMI do Windows Server 2022 Full otimizada para o Amazon EKS.

Kubernetes version 1.30

Kubernetes versão **1.30**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Inclui patches para o CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Atualizado containerd para 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes versão **1.29**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Atualizado containerd para 1.7.11. Atualizado kubelet para 1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Reconstrução do

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
				CNI e do `csi-proxy` usando o `golang 1.22.1`.
`1.29-2024.03.12`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.13`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.06`	`1.29.0`	`1.6.25`	`1.1.2`	Corrigido um erro em que a imagem de pausa era excluída incorretamente pelo processo de coleta de resíduos do `kubelet`.
`1.29-2024.01.09`	`1.29.0`	`1.6.18`	`1.1.2`	

Kubernetes version 1.28

Kubernetes versão **`1.28`**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
`1.28-2024.07.10`	`1.28.11`	`1.7.11`	`1.1.2`	Inclui patches para o `CVE-2024-5321` .
`1.28-2024.06.17`	`1.28.8`	`1.7.11`	`1.1.2`	Atualizado `containerd` para `1.7.11`.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.28.8.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes versão **1.27**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2024.07.10	1.27.15	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.27.12.
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	Inclui patches para o CVE-2023-3676 , CVE-2023-3893 e CVE-2023-3955 .
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	Foi corrigido o containers-roadmap problema #2042, que fazia com que os nós falhassem ao extrair imagens privadas do Amazon ECR.
1.27-2023.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes versão **1.26**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2024.07.10	1.26.15	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2024.06.17	1.26.15	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.26.15.
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.26-2024.03.12	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	Atualizado `containerd` para `1.6.18`. Atualizado `kubelet` para `1.26.9`. Foram adicionadas novas variáveis do ambiente de script de bootstrap (`SERVICE_IPV4_CIDR` e `EXCLUDED_SNAT_CIDRS`).
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Inclui patches para o `CVE-2023-3676`, `CVE-2023-3893` e `CVE-2023-3955`.
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Atualizado Kubernetes para 1.26.4. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes versão **1.25**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.25-2024.07.10	1.25.16	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.25-2024.06.17	1.25.16	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	Atualizado containerd para 1.6.28.
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.25-2024.03.12	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Atualizado

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
				`kubelet` para `1.25.14`. Foram adicionadas novas variáveis do ambiente de script de bootstrap (`SERVICE_IPV4_CIDR` e `EXCLUDED_SNAT_CIDRS`).
`1.25-2023.09.12`	`1.25.12`	`1.6.6`	`1.1.2`	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
`1.25-2023.08.17`	`1.25.12`	`1.6.6`	`1.1.2`	Inclui patches para o `CVE-2023-3676`, `CVE-2023-3893` e `CVE-2023-3955`.
`1.25-2023.08.08`	`1.25.9`	`1.6.6`	`1.1.1`	
`1.25-2023.07.11`	`1.25.9`	`1.6.6`	`1.1.1`	
`1.25-2023.06.20`	`1.25.9`	`1.6.6`	`1.1.1`	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
`1.25-2023.06.14`	`1.25.9`	`1.6.6`	`1.1.1`	Atualizado Kubernetes para `1.25.9`. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Instalação de um plug-in gMSA sem domínio para facilitar a autenticação gMSA para contêineres do Windows no Amazon EKS.
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes versão **1.24**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2024.07.10	1.24.17	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.24-2024.06.17	1.24.17	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	Atualizado containerd para 1.6.28.
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.24-2024.03.12	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Atualizado

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
				`kubelet` para `1.24.17`. Foram adicionadas novas variáveis do ambiente de script de bootstrap (`SERVICE_IPV4_CIDR` e `EXCLUDED_SNAT_CIDRS`).
`1.24-2023.09.12`	`1.24.16`	`1.6.6`	`1.1.2`	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
`1.24-2023.08.17`	`1.24.16`	`1.6.6`	`1.1.2`	Inclui patches para o `CVE-2023-3676` , `CVE-2023-3893` e `CVE-2023-3955` .
`1.24-2023.08.08`	`1.24.13`	`1.6.6`	`1.1.1`	
`1.24-2023.07.11`	`1.24.13`	`1.6.6`	`1.1.1`	
`1.24-2023.06.20`	`1.24.13`	`1.6.6`	`1.1.1`	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
`1.24-2023.06.14`	`1.24.13`	`1.6.6`	`1.1.1`	Atualizado Kubernetes para `1.24.13`. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Instalação de um plug-in gMSA sem domínio para facilitar a autenticação gMSA para contêineres do Windows no Amazon EKS.
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	A versão do Kubernetes sofreu um downgrad para a versão 1.24.7 porque a 1.24.10 tem um problema relatado no kube-proxy .
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.14	1.24.7	1.6.6	1.1.1	
1.24-2022.10.11	1.24.7	1.6.6	1.1.1	

AMI do Windows Server 2019 Core otimizada para o Amazon EKS

As tabelas a seguir listam as versões atuais e anteriores da AMI do Windows Server 2019 Core otimizada para o Amazon EKS.

Kubernetes version 1.30

Kubernetes versão **1.30**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Inclui patches para o CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Atualizado containerd para 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes versão **1.29**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Atualizado containerd para 1.7.11. Atualizado kubelet para 1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Corrigido um erro em que a imagem de pausa era excluída incorretamente pelo processo de coleta de resíduos do kubelet.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes versão **1.28**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.28.8.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes versão **1.27**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2024.07.10	1.27.15	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.27.12.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	Inclui patches para o CVE-2023-3676 , CVE-2023-3893 e CVE-2023-3955 .
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	Foi corrigido o containers-roadmap problema #2042, que fazia com que os nós falhassem ao extrair imagens privadas do Amazon ECR.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
11.27-2023.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes versão **1.26**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2024.07.10	1.26.15	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.26-2024.06.17	1.26.15	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.26.15.
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.26-2024.03.13	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Atualizado kubelet para 1.26.9. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Inclui patches para o CVE-2023-3676 , CVE-2023-3893 e CVE-2023-3955 .
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Atualizado Kubernetes para 1.26.4. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes versão **1.25**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.25-2024.07.10	1.25.16	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.25-2024.06.17	1.25.16	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	Atualizado containerd para 1.6.28.
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.25-2024.03.13	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Atualizado

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
				`kubelet` para `1.25.14`. Foram adicionadas novas variáveis do ambiente de script de bootstrap (`SERVICE_IPV4_CIDR` e `EXCLUDED_SNAT_CIDRS`).
`1.25-2023.09.12`	`1.25.12`	`1.6.6`	`1.1.2`	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
`1.25-2023.08.17`	`1.25.12`	`1.6.6`	`1.1.2`	Inclui patches para o `CVE-2023-3676` , `CVE-2023-3893` e `CVE-2023-3955` .
`1.25-2023.08.08`	`1.25.9`	`1.6.6`	`1.1.1`	
`1.25-2023.07.11`	`1.25.9`	`1.6.6`	`1.1.1`	
`1.25-2023.06.20`	`1.25.9`	`1.6.6`	`1.1.1`	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
`1.25-2023.06.14`	`1.25.9`	`1.6.6`	`1.1.1`	Atualizado Kubernetes para `1.25.9`. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Instalação de um plug-in gMSA sem domínio para facilitar a autenticação gMSA para contêineres do Windows no Amazon EKS.
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes versão **1.24**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2024.07.10	1.24.17	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.24-2024.06.17	1.24.17	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	Atualizado containerd para 1.6.28.
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.24-2024.03.13	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Atualizado

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
				`kubelet` para `1.24.17`. Foram adicionadas novas variáveis do ambiente de script de bootstrap (`SERVICE_IPV4_CIDR` e `EXCLUDED_SNAT_CIDRS`).
`1.24-2023.09.12`	`1.24.16`	`1.6.6`	`1.1.2`	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
`1.24-2023.08.17`	`1.24.16`	`1.6.6`	`1.1.2`	Inclui patches para o `CVE-2023-3676` , `CVE-2023-3893` e `CVE-2023-3955` .
`1.24-2023.08.08`	`1.24.13`	`1.6.6`	`1.1.1`	
`1.24-2023.07.11`	`1.24.13`	`1.6.6`	`1.1.1`	
`1.24-2023.06.20`	`1.24.13`	`1.6.6`	`1.1.1`	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
`1.24-2023.06.14`	`1.24.13`	`1.6.6`	`1.1.1`	Atualizado Kubernetes para `1.24.13`. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Instalação de um plug-in gMSA sem domínio para facilitar a autenticação gMSA para contêineres do Windows no Amazon EKS.
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	A versão do Kubernetes sofreu um downgrad para a versão 1.24.7 porque a 1.24.10 tem um problema relatado no kube-proxy .
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.13	1.24.7	1.6.6	1.1.1	
1.24-2022.11.08	1.24.7	1.6.6	1.1.1	

AMI completa do Windows Server 2019 Core otimizada para Amazon EKS

As tabelas a seguir listam as versões atuais e anteriores da AMI do Windows Server 2019 Full otimizada para o Amazon EKS.

Kubernetes version 1.30

Kubernetes versão **1.30**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Inclui patches para o CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Atualizado containerd para 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes versão **1.29**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Atualizado containerd para 1.7.11. Atualizado kubelet para 1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Corrigido um erro em que a imagem de pausa era excluída incorretamente pelo processo de coleta de resíduos do kubelet.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes versão **1.28**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.28.8.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes versão **1.27**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2024.07.10	1.27.15	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.27.12.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	Inclui patches para o CVE-2023-3676 , CVE-2023-3893 e CVE-2023-3955 .
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	Foi corrigido o containers-roadmap problema #2042, que fazia com que os nós falhassem ao extrair imagens privadas do Amazon ECR.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.27-2023.05.17	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes versão **1.26**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2024.07.10	1.26.15	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.26-2024.06.17	1.26.15	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.26.15.
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.26-2024.03.13	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Atualizado kubelet para 1.26.9. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Inclui patches para o CVE-2023-3676 , CVE-2023-3893 e CVE-2023-3955 .
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Atualizado Kubernetes para 1.26.4. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes versão **1.25**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.25-2024.07.10	1.25.16	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.25-2024.06.17	1.25.16	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	Atualizado containerd para 1.6.28.
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.25-2024.03.13	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Atualizado

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
				`kubelet` para `1.25.14`. Foram adicionadas novas variáveis do ambiente de script de bootstrap (`SERVICE_IPV4_CIDR` e `EXCLUDED_SNAT_CIDRS`).
`1.25-2023.09.12`	`1.25.12`	`1.6.6`	`1.1.2`	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
`1.25-2023.08.17`	`1.25.12`	`1.6.6`	`1.1.2`	Inclui patches para o `CVE-2023-3676`, `CVE-2023-3893` e `CVE-2023-3955`.
`1.25-2023.08.08`	`1.25.9`	`1.6.6`	`1.1.1`	
`1.25-2023.07.11`	`1.25.9`	`1.6.6`	`1.1.1`	
`1.25-2023.06.20`	`1.25.9`	`1.6.6`	`1.1.1`	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
`1.25-2023.06.14`	`1.25.9`	`1.6.6`	`1.1.1`	Atualizado Kubernetes para `1.25.9`. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Instalação de um plug-in gMSA sem domínio para facilitar a autenticação gMSA para contêineres do Windows no Amazon EKS.
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes versão **1.24**

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2024.07.10	1.24.17	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.24-2024.06.17	1.24.17	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	Atualizado containerd para 1.6.28.
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução do CNI e do csi-proxy usando o golang 1.22.1.
1.24-2024.03.13	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Atualizado

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
				`kubelet` para `1.24.17`. Foram adicionadas novas variáveis do ambiente de script de bootstrap (`SERVICE_IPV4_CIDR` e `EXCLUDED_SNAT_CIDRS`).
`1.24-2023.09.12`	`1.24.16`	`1.6.6`	`1.1.2`	Atualizou o plug-in CNI do Amazon VPC para usar Kubernetes o binário do conector, que obtém Pod o endereço IP do servidor da Kubernetes API. Solicitação pull mesclada #100.
`1.24-2023.08.17`	`1.24.16`	`1.6.6`	`1.1.2`	Inclui patches para o `CVE-2023-3676` , `CVE-2023-3893` e `CVE-2023-3955` .
`1.24-2023.08.08`	`1.24.13`	`1.6.6`	`1.1.1`	
`1.24-2023.07.11`	`1.24.13`	`1.6.6`	`1.1.1`	
`1.24-2023.06.21`	`1.24.13`	`1.6.6`	`1.1.1`	Resolvido um problema que fazia com que a lista de pesquisa de sufixos de DNS fosse preenchida incorretamente.
`1.24-2023.06.14`	`1.24.13`	`1.6.6`	`1.1.1`	Atualizado Kubernetes para `1.24.13`. Adicionado suporte para mapeamento de porta de host em CNI. Solicitação pull mesclada #93.

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	Corrigido um bug que causava o problema de conectividade de rede #1126 em pods após a reinicialização do nó. Introdução de um novo parâmetro de configuração de script de bootstrap (ExcludedSnatCIDRs).
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	Mecanismo de recuperação adicionado para kubelet e kube-proxy em caso de falha de serviço.
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Instalação de um plug-in gMSA sem domínio para facilitar a autenticação gMSA para contêineres do Windows no Amazon EKS.
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	A versão do Kubernetes sofreu um downgrad para a versão 1.24.7 porque a 1.24.10 tem um problema relatado no kube-proxy .
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	

AMI version	Versão do **kubelet**	Versão do **containerd**	Versão do **csi-proxy**	Notas de release
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.14	1.24.7	1.6.6	1.1.1	
1.24-2022.10.12	1.24.7	1.6.6	1.1.1	

Recuperar IDs de AMI do Microsoft Windows recomendadas

Você pode recuperar programaticamente o ID da imagem de máquina da Amazon (AMI) para AMIs otimizadas para o Amazon EKS consultando a API Repositório de parâmetros do AWS Systems Manager. Esse parâmetro elimina a necessidade de pesquisar manualmente IDs de AMIs otimizadas para o Amazon EKS. Para obter mais informações sobre a API Systems Manager Parameter Store, consulte GetParameter. A entidade principal do IAM que você usou deve ter a permissão `ssm:GetParameter` do IAM para recuperar os metadados da AMI otimizada do Amazon EKS.

Você pode recuperar o ID de imagem da mais recente AMI do Windows otimizada recomendada para o Amazon EKS com o comando a seguir, que usa o subparâmetro `image_id`. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado:

- Substitua *`release`* por uma das opções a seguir.
  - Use `2022` para o Windows Server 2022, mas somente se você estiver usando a versão `1.24` ou posterior do Kubernetes.
  - Use `2019` para o Windows Server 2019.
- Substitua *`installation-option`* por uma das opções a seguir. Para obter mais informações, consulte O que é a opção de instalação Server Core no Windows Server?.
  - Use `Core` para uma instalação mínima com uma superfície de ataque menor.
  - Use `Full` para incluir a experiência de área de trabalho do Windows.
- Substitua *`kubernetes-version`* por qualquer versão compatível do Amazon EKS.

- Substitua *region-code* por uma Região da AWS do Amazon EKS compatível para a qual você deseja o ID da AMI.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-release-
English-installation-option-EKS_Optimized-kubernetes-version/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Aqui está um exemplo de comando após as substituições do espaço reservado terem sido feitas.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2022-
English-Core-EKS_Optimized-1.30/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

Veja um exemplo de saída abaixo.

```
ami-1234567890abcdef0
```

Crie uma AMI do Windows personalizada com o Image Builder

Você pode usar o EC2 Image Builder para criar AMIs Windows personalizadas otimizadas para Amazon EKS com uma das seguintes opções:

- Usar uma AMI Windows otimizada para Amazon EKS como base
- Usar o componente de compilação gerenciado pela Amazon

Com ambos os métodos, você deve criar sua própria fórmula do Image Builder. Para obter mais informações, consulte Criar uma nova versão de uma fórmula de imagem, no Guia do usuário do Image Builder.

> **Important**
>
> Os seguintes componentes gerenciados pela Amazon para o `eks` incluem patches para o `CVE-2024-5321`.
>
> - `1.24.5` e superior
> - `1.25.4` e superior
> - `1.26.4` e superior

- `1.27.2` e superior
- `1.28.2` e superior
- `1.29.2` e superior
- `1.30.1` e superior

Usar uma AMI Windows otimizada para Amazon EKS como base

Essa opção é a maneira recomendada de criar AMIs Windows personalizadas. As AMIs Windows otimizadas para Amazon EKS que fornecemos são atualizadas de maneira mais frequente do que o componente de compilação gerenciado pela Amazon.

1. Inicie uma fórmula do Image Builder.
   a. Abra o console do EC2 Image Builder em https://console.aws.amazon.com/imagebuilder.
   b. No painel de navegação esquerdo, escolha Image recipes (Fórmulas de imagens).
   c. Escolha Create image recipe (Criar fórmula de imagem).
2. Na seção Recipe details (Detalhes da fórmula), insira um nome e uma versão.
3. Especifique o ID da AMI Windows otimizada para Amazon EKS na seção Base image (Imagem base).
   a. Escolha Enter custom AMI ID (Inserir ID de AMI personalizada).
   b. Recupere o ID da AMI da versão do SO Windows necessária. Para obter mais informações, consulte Recuperar IDs de AMI do Microsoft Windows recomendadas.
   c. Em AMI ID, insira o ID da AMI personalizado. Se o ID da AMI não for encontrado, certifique-se de que a Região da AWS do ID da AMI corresponda à Região da AWS mostrada no canto superior direito do console.
4. (Opcional) Para obter as atualizações de segurança mais recentes, adicione o componente `update-windows` na seção Build components - (Componentes de compilação).
   a. Na lista suspensa à direita da caixa de pesquisa Find components by name (Localizar componentes por nome), escolha Amazon-managed (Gerenciado pela Amazon).
   b. Na caixa Find components by name (Localizar componentes por nome), insira **`update-windows`**.
   c. Marque a caixa de seleção do resultado da pesquisa de **`update-windows`**. Esse componente inclui os patches do Windows mais recentes para o sistema operacional.

5. Conclua as entradas restantes da fórmula de imagem com as configurações necessárias. Para obter mais informações, consulte Criar uma nova versão de fórmula de imagem (console), no Guia do usuário do Image Builder.
6. Escolha Create recipe (Criar fórmula).
7. Use a nova fórmula de imagem em um pipeline de imagens novo ou existente. Quando o pipeline de imagens for executado com êxito, a AMI personalizada será listada como imagem de saída e estará pronta para uso. Para obter mais informações, consulte Criar um pipeline de imagem usando o assistente do console do EC2 Image Builder.

Usar o componente de compilação gerenciado pela Amazon

Quando o uso de uma AMI Windows otimizada para Amazon EKS como base não for viável, você poderá usar o componente de compilação gerenciado pela Amazon. Essa opção pode estar atrasada em relação às versões do Kubernetes mais recentes compatíveis.

1. Inicie uma fórmula do Image Builder.
   a. Abra o console do EC2 Image Builder em https://console.aws.amazon.com/imagebuilder.
   b. No painel de navegação esquerdo, escolha Image recipes (Fórmulas de imagens).
   c. Escolha Create image recipe (Criar fórmula de imagem).
2. Na seção Recipe details (Detalhes da fórmula), insira um nome e uma versão.
3. Determine qual opção usar para criar sua AMI personalizada na seção Base image (Imagem base):
   - Select managed images (Selecionar imagens gerenciadas): escolha Windows em Image Operating System (OS) (Sistema operacional da imagem). Em seguida, escolha uma das seguintes opções para Image origin (Origem da imagem).
     - Quick start (Amazon-managed) (Início rápido, gerenciado pela Amazon): na lista suspensa Imagem name (Nome da imagem), selecione uma versão compatível do Windows Server compatível com o Amazon EKS. Para obter mais informações, consulte Criar nós com AMIs do Windows otimizadas.
     - Images owned by me (Imagens que pertencem a mim): para Image name (Nome da imagem), escolha o ARN da sua própria imagem com sua própria licença. A imagem que você fornece ainda não pode ter componentes do Amazon EKS instalados.

- Enter custom AMI ID (Inserir ID da AMI personalizado): para o ID da AMI, insira o ID da AMI com sua própria licença. A imagem que você fornece ainda não pode ter componentes do Amazon EKS instalados.

4. Na seção Build components - Windows (Componentes de compilação - Windows), faça o seguinte:

   a. Na lista suspensa à direita da caixa de pesquisa Find components by name (Localizar componentes por nome), escolha Amazon-managed (Gerenciado pela Amazon).

   b. Na caixa Find components by name (Localizar componentes por nome), insira **`eks`**.

   c. Marque a caixa de seleção do resultado da pesquisa de **`eks-optimized-ami-windows`**, mesmo que o resultado retornado não seja a versão desejada.

   d. Na caixa Find components by name (Localizar componentes por nome), insira **`update-windows`**.

   e. Marque a caixa de seleção do resultado da pesquisa de update-windows. Esse componente inclui os patches do Windows mais recentes para o sistema operacional.

5. Na seção Selected components (Componentes selecionados), faça o seguinte:

   a. Escolha opções de versionamento para **`eks-optimized-ami-windows`**.

   b. Escolha Specify component version (Especificar versão do componente).

   c. No campo Component Version (Versão do componente), insira ***`version.x`*** , substituindo *`version`* por uma versão compatível do Kubernetes. Inserir um *`x`* como parte do número da versão indica o uso da versão mais recente do componente que também se alinha à parte da versão que você define explicitamente. Preste atenção na saída do console, pois ela informará se a versão desejada está disponível como componente gerenciado. Lembre-se de que as versões do Kubernetes mais recentes podem não estar disponíveis para o componente de compilação. Para obter mais informações sobre as versões disponíveis, consulte Recuperar informações sobre versões de componentes `eks-optimized-ami-windows`.

      > **Note**
      >
      > As seguintes versões do componente de compilação `eks-optimized-ami-windows` exigem uma versão de `eksctl` `0.129` ou inferior:
      >
      > - `1.24.0`

6. Conclua as entradas restantes da fórmula de imagem com as configurações necessárias. Para obter mais informações, consulte Criar uma nova versão de fórmula de imagem (console), no Guia do usuário do Image Builder.
7. Escolha Create recipe (Criar fórmula).
8. Use a nova fórmula de imagem em um pipeline de imagens novo ou existente. Quando o pipeline de imagens for executado com êxito, a AMI personalizada será listada como imagem de saída e estará pronta para uso. Para obter mais informações, consulte Criar um pipeline de imagem usando o assistente do console do EC2 Image Builder.

Recuperar informações sobre versões de componentes **`eks-optimized-ami-windows`**

É possível recuperar informações específicas sobre o que está instalado com cada componente. Por exemplo, você pode verificar qual versão do `kubelet` está instalada. Os componentes passam por testes funcionais nas versões do sistema operacional Windows compatíveis com o Amazon EKS. Para obter mais informações, consulte Calendário de lançamento. Quaisquer outras versões do sistema operacional Windows selecionadas podem não ser compatíveis com o componente.

1. Abra o console do EC2 Image Builder em https://console.aws.amazon.com/imagebuilder.
2. No painel de navegação à esquerda, escolha Components (Componentes).
3. Na lista suspensa à direita da caixa de pesquisa Find components by name (Localizar componentes por nome), altere Owned by me (Pertencem a mim) para Quick start (Amazon-managed) (Início rápido, gerenciado pela Amazon).
4. Na caixa Find components by name (Localizar componentes por nome), insira **`eks`**.
5. (Opcional) Se estiver usando uma versão recente, classifique a coluna Version (Versão) em ordem decrescente, escolhendo-a duas vezes.
6. Escolha o link **`eks-optimized-ami-windows`** com a versão desejada.

As informações em Description (Descrição) na página resultante mostra as informações específicas.

Armazenar dados de aplicações para seu cluster

Este capítulo aborda as opções de armazenamento para clusters do Amazon EKS.

Tópicos

Armazene volumes do Kubernetes com o Amazon EBS

O driver de Container Storage Interface (CSI) do Amazon Elastic Block Store (Amazon EBS) gerencia o ciclo de vida dos volumes do Amazon EBS como armazenamento para os volumes do Kubernetes que você criar. O driver da CSI do Amazon EBS cria volumes do Amazon EBS para estes tipos de volumes do Kubernetes: volumes temporários e volumes persistentes genéricos.

Considerações

- Você não pode montar volumes do Amazon EBS em Pods do Fargate.
- Você pode executar o controlador da CSI do Amazon EBS em nós do Fargate, mas o DaemonSet do nó da CSI do Amazon EBS só pode ser executado em instâncias do Amazon EC2.

> **Important**
>
> Para utilizar a funcionalidade de snapshot do driver de CSI do Amazon EBS, é necessário instalar o criador de snapshots externo antes ou depois da instalação do complemento. Os componentes do snapshotter externo devem ser instalados na seguinte ordem:

- `CustomResourceDefinition` (CRD) para `volumesnapshotclasses`, `volumesnapshots` e `volumesnapshotcontents`
- RBAC (`ClusterRole`, `ClusterRoleBinding` e assim por diante)
- implantação do controlador

Para obter mais informações, consulte Snapshotter de CSI no GitHub.

Pré-requisitos

- Um cluster existente. Para ver a versão necessária da plataforma, execute o comando a seguir.

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- Um provedor OpenID Connect (OIDC) do AWS Identity and Access Management (IAM) existente para o cluster. Para determinar se você tem ou para criar uma, consulte Criar um provedor OIDC do IAM para o cluster.
- Se você estiver usando uma `PodSecurityPolicy` no âmbito do cluster, certifique-se de que o complemento tenha permissões suficientes para ser implantado. Para obter as permissões exigidas por cada Pod de complemento, consulte a definição relevante de manifesto de complemento no GitHub.

Etapa 1: Criar uma função do IAM

O plugin CSI do Amazon EBS requer permissões do IAM para fazer chamadas para APIs da AWS em seu nome. Se você não concluir essas etapas, a tentativa de instalar o complemento e de executar `kubectl describe pvc` mostrarão `failed to provision volume with StorageClass` junto com erro `could not create volume in EC2: UnauthorizedOperation`. Para obter mais informações, consulte Set up driver permission(Configurar permissão de driver) no GitHub.

> **Note**
>
> Pods terão acesso às permissões atribuídas ao perfil do IAM, a menos que você bloqueie o acesso ao IMDS. Para obter mais informações, consulte Práticas recomendadas de segurança para o Amazon EKS.

O procedimento a seguir mostra como criar um perfil do IAM e associar a política gerenciada da AWS a ele. É possível usar o `eksctl`, o AWS Management Console ou a AWS CLI.

> **Note**
>
> As etapas específicas desse procedimento foram escritas para usar o driver como um complemento do Amazon EKS. Diferentes etapas para usar o driver como um complemento autogerenciado. Para obter mais informações, consulte Configurar permissão de driver no GitHub.

eksctl

Para criar o perfil do IAM do plugin Amazon EBS CSI com o **`eksctl`**

1. Criar um perfil do IAM e associar uma política. AWS mantém uma política AWS gerenciada ou você pode criar sua própria política personalizada. Você pode criar uma função IAM e anexar a política gerenciada AWS com o seguinte comando. Substitua o *`my-cluster`* pelo nome do cluster. O comando implanta uma pilha do AWS CloudFormation que cria um perfil do IAM e anexa a política do IAM a ele. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
eksctl create iamserviceaccount \
        --name ebs-csi-controller-sa \
        --namespace kube-system \
        --cluster my-cluster \
        --role-name AmazonEKS_EBS_CSI_DriverRole \
        --role-only \
        --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicy \
        --approve
```

2. Se você usar uma chave do KMS personalizada para criptografia em seus volumes do Amazon EBS, personalize o perfil do IAM conforme necessário. Por exemplo, faça o seguinte:

   a. Copie e cole o código a seguir em um novo arquivo *kms-key-for-encryption-on-ebs*`.json`. Substitua *custom-key-arn* pelo ARN da chave do KMS personalizada.

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["custom-key-arn"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["custom-key-arn"]
        }
      ]
    }
```

   b. Crie a política . Você pode alterar *KMS_Key_For_Encryption_On_EBS_Policy* para um nome diferente. Porém, se fizer isso, não se esqueça de alterá-lo também em etapas posteriores.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Anexe a política do IAM necessária à função com o comando a seguir. Substitua *111122223333* pelo ID da sua conta. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
aws iam attach-role-policy \
      --policy-arn
 arn:aws:iam::111122223333:policy/KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

AWS Management Console

Para criar a função do IAM do plugin Amazon EBS CSI com o AWS Management Console

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.
3. Na página Roles (Funções), selecione Create role (Criar função).
4. Na página Select trusted entity (Selecionar entidade confiável), faça o seguinte:
   a. Na seção Trusted entity type (Tipo de entidade confiável), escolha Web identity (Identidade da Web).
   b. Em Identity provider (Provedor de identidade), escolha OpenID ConnectOpenID Connect provider URL (URL do provedor OpenID Connect) para o cluster (conforme mostrado na guia Overview [Visão geral] do Amazon EKS).
   c. Para Audience (Público), escolha `sts.amazonaws.com`.
   d. Escolha Próximo.
5. Na página Add permissions (Adicionar permissões), faça o seguinte:
   a. Na caixa Filter policies (Políticas de filtro) insira `AmazonEBSCSIDriverPolicy`.

b. Marque a caixa de seleção à esquerda do `AmazonEBSCSIDriverPolicy` retornado na pesquisa.

c. Escolha Próximo.

6. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:

a. Em Role name (Nome da função), insira um nome exclusivo para a função, como ***AmazonEKS_EBS_CSI_DriverRole***.

b. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.

c. Selecione Criar função.

7. Depois que a função for criada, escolha a função no console a fim de abri-la para edição.

8. Escolha a guia Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).

9. Encontre a linha semelhante à seguinte:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

Adicione uma vírgula ao final da linha anterior e depois adicione a linha a seguir após a anterior. Substitua *region-code* pela Região da AWS em que está o cluster. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* pelo ID do provedor OIDC do cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:ebs-csi-controller-sa"
```

10. Escolha Update Policy (Atualizar política) para concluir.

11. Se você usar uma chave do KMS personalizada para criptografia em seus volumes do Amazon EBS, personalize o perfil do IAM conforme necessário. Por exemplo, faça o seguinte:

a. No painel de navegação à esquerda, escolha Políticas.

b. Na página Policies (Políticas), escolha Create Policy (Criar política).

c. Na página Criar política, escolha a guia JSON.

d. Copie e cole o seguinte código no editor, substituindo *custom-key-arn* pelo ARN da chave do KMS personalizada:

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["custom-key-arn"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["custom-key-arn"]
        }
      ]
    }
```

e. Escolha Próximo: etiquetas.

f. Na página Add tags (optional) (Adicionar etiquetas - opcional), escolha Next: Review (Próximo: revisar).

g. Em Name (Nome), insira um nome exclusivo para a sua política (por exemplo, ***KMS_Key_For_Encryption_On_EBS_Policy***).

h. Escolha Criar política.

i. No painel de navegação à esquerda, escolha Roles.

j. Escolha ***AmazonEKS_EBS_CSI_DriverRole*** no console para abri-lo para edição.

k. Na lista suspensa Add permissions (Adicionar permissões), escolha Attach policies (Anexar políticas).

l. Na caixa Filter policies (Políticas de filtro) insira *KMS_Key_For_Encryption_On_EBS_Policy*.

m. Marque a caixa de seleção à esquerda do *KMS_Key_For_Encryption_On_EBS_Policy* retornado na pesquisa.

n. Escolha Anexar políticas.

AWS CLI

Para criar a função do IAM do plugin Amazon EBS CSI com o AWS CLI

1. Exiba a URL do provedor OIDC do cluster. Substitua *my-cluster* pelo nome do cluster. Se o resultado do comando for `None`, revise os Pré-requisitos.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

Veja um exemplo de saída abaixo.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Crie o perfil do IAM, concedendo a ação `AssumeRoleWithWebIdentity`.

a. Copie o conteúdo a seguir em um arquivo chamado *aws-ebs-csi-driver-trust-policy*`.json`. Substitua *111122223333* pelo ID da sua conta. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* e *region-code* pelos valores retornados na etapa anterior. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
```

```
            "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
          },
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {
            "StringEquals": {
              "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
              "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:ebs-csi-controller-sa"
            }
          }
        }
      ]
    }
```

b. Crie a função. Você pode alterar *AmazonEKS_EBS_CSI_DriverRole* para um nome diferente. Se você alterá-lo, não se esqueça de alterá-lo em etapas superiores.

```
aws iam create-role \
      --role-name AmazonEKS_EBS_CSI_DriverRole \
      --assume-role-policy-document file://"aws-ebs-csi-driver-trust-
policy.json"
```

3. Associar uma política. AWS mantém uma política AWS gerenciada ou você pode criar sua própria política personalizada. Anexe a política gerenciada pela AWS função com o comando a seguir. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
aws iam attach-role-policy \
      --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
 \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

4. Se você usar uma chave do KMS personalizada para criptografia em seus volumes do Amazon EBS, personalize o perfil do IAM conforme necessário. Por exemplo, faça o seguinte:

a. Copie e cole o código a seguir em um novo arquivo *kms-key-for-encryption-on-ebs*.json. Substitua *custom-key-arn* pelo ARN da chave do KMS personalizada.

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["custom-key-arn"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["custom-key-arn"]
        }
      ]
    }
```

b. Crie a política . Você pode alterar *KMS_Key_For_Encryption_On_EBS_Policy* para um nome diferente. Porém, se fizer isso, não se esqueça de alterá-lo também em etapas posteriores.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Anexe a política do IAM necessária à função com o comando a seguir. Substitua *111122223333* pelo ID da sua conta. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
aws iam attach-role-policy \
      --policy-arn
 arn:aws:iam::111122223333:policy/KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

Agora que você criou o perfil do IAM do driver de CSI do Amazon EBS, poderá avançar para a próxima seção. Quando você implanta o complemento com esse perfil do IAM, ele cria e é configurado para usar uma conta de serviço chamada `ebs-csi-controller-sa`. A conta de serviço está vinculada a uma `clusterrole` Kubernetes que recebe as permissões necessárias do Kubernetes.

Etapa 2: obter o driver de CSI do Amazon EBS

Recomendamos que você instale o driver de CSI do Amazon EBS por meio do complemento do Amazon EKS para reforçar a segurança e reduzir o volume do trabalho. Para adicionar um complemento do Amazon EKS ao cluster, consulte Criar um complemento do Amazon EKS. Para obter mais informações sobre complementos, consulte Complementos do Amazon EKS.

> **Important**
>
> Antes de adicionar o driver do Amazon EBS como um complemento do Amazon EKS, verifique se você não tem uma versão autogerenciada do driver instalada no seu cluster. Em caso afirmativo, consulte Desinstalar um driver de CSI autogerenciado do Amazon EBS em GitHub.

Como alternativa, se desejar uma instalação autogerenciada do driver de CSI do Amazon EFS, consulte Instalação em GitHub.

Etapa 3: implantar uma aplicação de exemplo

Você pode implantar toda uma variedade de aplicações de exemplo e modificá-las conforme necessário. Para obter mais informações, consulte Exemplos de Kubernetes no GitHub.

Perguntas frequentes sobre migração de CSI do Amazon EBS

> **Important**
>
> Se o Pods estiver sendo executado em um cluster da versão `1.22` ou anterior, será necessário instalar o driver da CSI do Amazon EBS antes de atualizar seu cluster para a versão `1.23` para evitar a interrupção do serviço.

O recurso de migração da interface de armazenamento em contêiner (CSI) do Amazon EBS transfere a responsabilidade de lidar com as operações de armazenamento do provisionador de armazenamento de EBS “em árvore” do Amazon EBS para o driver da CSI do Amazon EBS.

O que são drivers de CSI?

Drivers de CSI:

- Substituem os drivers de armazenamento “em árvore” do Kubernetes que existem no código-fonte do projeto Kubernetes.
- Funcionam com provedores de armazenamento, como o Amazon EBS.
- fornecem um modelo simplificado de plug-in que torna mais fácil para provedores de armazenamento como a AWS lançar recursos e manter a compatibilidade sem depender do ciclo de versões do Kubernetes.

Para obter mais informações consulte Introduction (Introdução) na documentação de CSI do Kubernetes.

O que é migração de CSI?

O recurso migração de CSI do Kubernetes transfere a responsabilidade de processamento das operações de armazenamento dos plug-ins de armazenamento em árvore existentes, como `kubernetes.io/aws-ebs`, para os drivers de CSI correspondentes. Os objetos `StorageClass`,

`PersistentVolume` e `PersistentVolumeClaim` (PVC) existentes continuam funcionando, desde que o driver de CSI correspondente esteja instalado. Quando o recurso estiver habilitado:

- As workloads existentes que utilizarem PVCs continuarão funcionando como sempre.
- O Kubernetes transmitirá o controle de todas as operações de gerenciamento de armazenamento para os drivers de CSI.

Para obter mais informações, consulte Kubernetes`1.23`: Kubernetes In-Tree to CSI Volume Migration Status Update (Atualização do status de migração de volumes CSI) no blog do Kubernetes.

Para ajudar você a migrar do plug-in em árvore para os drivers de CSI, os sinalizadores `CSIMigration` e `CSIMigrationAWS` são habilitados por padrão em clusters versão `1.23` e posteriores do Amazon EKS. Esses sinalizadores permitem que o cluster traduza as APIs em árvore para suas APIs equivalentes do driver de CSI. Esses sinalizadores estão definidos no ambiente de gerenciamento do Kubernetes gerenciado pelo Amazon EKS e nas configurações `kubelet` definidas nas AMIs otimizadas para o Amazon EKS. Se você tiver Pods usando volumes do Amazon EBS em seu cluster, será necessário instalar o driver de CSI do Amazon EBS antes de atualizar o cluster para a versão **`1.23`**. Caso contrário, talvez as operações de volume (p. ex., provisionamento e montagem) não funcionem conforme o esperado. Para obter mais informações, consulte Armazene volumes do Kubernetes com o Amazon EBS.

> **Note**
>
> O provisionador `StorageClass` em árvore tem o nome `kubernetes.io/aws-ebs`. O provisionador `StorageClass` de CSI do Amazon EBS tem o nome `ebs.csi.aws.com`.

Posso montar volumes da **`kubernetes.io/aws-ebs StorageClass`** em clusters da versão **`1.23`** e posteriores?

Sim, desde que o driver de CSI do Amazon EBS esteja instalado. Para clusters `1.23` e posteriores recém-criados, recomendamos a instalação do driver de CSI do Amazon EBS como parte do processo de criação do cluster. Também recomendamos usar apenas `StorageClasses` com base no provisionador `ebs.csi.aws.com`.

Se você tiver atualizado o ambiente de gerenciamento do cluster para a versão `1.23` e ainda não tiver atualizado seus nós para a versão `1.23`, os sinalizadores `CSIMigration` e

`CSIMigrationAWS` do `kubelet` não estarão habilitados. Nesse caso, o driver em árvore é usado para montar volumes baseados em `kubernetes.io/aws-ebs`. No entanto, o driver de CSI do Amazon EBS ainda deverá estar instalado para garantir que seja possível agendar Pods usando volumes baseados em `kubernetes.io/aws-ebs`. O driver também é necessário para que outras operações de volume sejam bem-sucedidas.

Posso provisionar volumes da **`kubernetes.io/aws-ebs StorageClass`** em clusters **`1.23`** e posteriores do Amazon EKS?

Sim, desde que o driver de CSI do Amazon EBS esteja instalado.

O provisionador **`kubernetes.io/aws-ebs StorageClass`** será removido do Amazon EKS algum dia?

Não há mais suporte para o provisionador `kubernetes.io/aws-ebs` `StorageClass` e para o tipo de volume `awsElasticBlockStore`, mas não há planos para removê-los. Esses recursos são tratados como parte da API do Kubernetes.

Como instalo o driver de CSI do Amazon EBS?

Recomendamos instalar o Complemento do Amazon EKS do driver da CSI do Amazon EBS. Quando for necessário executar uma atualização no complemento do Amazon EKS, você iniciará a atualização e o Amazon EKS fará a atualização do complemento para você. Para gerenciar o driver por conta própria, você poderá instalá-lo usando o chart do Helm de código aberto.

> **Important**
>
> O driver do Amazon EBS para Kubernetes em árvore é executado no ambiente de gerenciamento do Kubernetes. Ele usa permissões do IAM atribuídas ao Função do IAM do cluster do Amazon EKS para provisionar volumes do Amazon EBS. O driver de CSI do Amazon EBS funciona em nós. O driver precisa de permissões do IAM para provisionar volumes. Para obter mais informações, consulte Etapa 1: Criar uma função do IAM.

Como verifico se o driver de CSI do Amazon EBS está instalado no meu cluster?

Para determinar se o driver está instalado em seu cluster, execute o seguinte comando:

```
kubectl get csidriver ebs.csi.aws.com
```

Para verificar se essa instalação é gerenciada pelo Amazon EKS, execute o seguinte comando:

```
aws eks list-addons --cluster-name my-cluster
```

O Amazon EKS impedirá uma atualização de cluster para a versão **1.23** se eu ainda não tiver instalado o driver de CSI do Amazon EBS?

Nº

E se eu esquecer de instalar o driver de CSI do Amazon EBS antes de atualizar meu cluster para a versão 1.23? Posso instalar o driver depois de atualizar meu cluster?

Sim, mas as operações de volume que exigirem o driver de CSI do Amazon EBS falharão após a atualização do cluster até que o driver seja instalado.

Qual é a **StorageClass** padrão aplicada em clusters versão **1.23** e posteriores recém-criados do Amazon EKS?

O comportamento padrão de `StorageClass` permanece inalterado. Com cada novo cluster, o Amazon EKS aplica um `StorageClass` chamado `gp2` baseado em `kubernetes.io/aws-ebs`. Não planejamos remover essa `StorageClass` de clusters recém-criados. Além da `StorageClass` padrão do cluster, se você criar uma `StorageClass` baseada em `ebs.csi.aws.com` sem especificar um tipo de volume, o driver de CSI do Amazon EBS usará `gp3` como padrão.

O Amazon EKS fará alguma alteração nas **StorageClasses** já presentes em meu cluster existente quando eu atualizar meu cluster para a versão **1.23**?

Nº

Como migro um volume persistente da `StorageClasskubernetes.io/aws-ebs` para `ebs.csi.aws.com` usando snapshots?

Para migrar um volume persistente, consulte Migrating Amazon EKS clusters from gp2 to gp3 EBS volumes (Migração de clusters do Amazon EKS de volumes gp2 para gp3 do EBS) no blog da AWS.

Como faço para modificar um volume do Amazon EBS usando anotações?

Começando com o `aws-ebs-csi-driver` `v1.19.0-eksbuild.2`, é possível modificar volumes do Amazon EBS usando anotações em seus`PersistentVolumeClaims` (PVC). O novo recurso de modificação de volume é implementado como um sidecar adicional chamado `volumemodifier`. Para obter mais informações, consulte Simplificar a migração e a modificação de volumes do Amazon EBS no Kubernetes usando o driver da CSI do EBS no blog da AWS.

Há compatibilidade para a migração de workloads do Windows?

Sim. Se você estiver instalando o driver de CSI do Amazon EBS usando o chart do Helm de código aberto, defina `node.enableWindows` como `true`. Isso é definido por padrão ao instalar o driver de CSI do Amazon EBS como complemento do Amazon EKS. Ao criar `StorageClasses`, defina `fsType` com um sistema de arquivos do Windows, como `ntfs`. Em seguida, as operações de volume para workloads do Windows serão migradas para o driver de CSI do Amazon EBS da mesma forma que seriam para workloads do Linux.

Armazenar um sistema de arquivos elástico com o Amazon EFS

O Amazon Elastic File System (Amazon EFS) fornece armazenamento de arquivos sem servidor e totalmente elástico para que você possa compartilhar dados de arquivos sem provisionar ou gerenciar a capacidade e a performance do armazenamento. O driver Container Storage Interface (CSI) do Amazon EFS fornece uma interface CSI que permite que os clusters do Kubernetes executados na AWS gerenciem o ciclo de vida dos sistemas de arquivos do Amazon EFS. Este tópico mostra como implantar o driver da CSI do Amazon EFS no cluster do Amazon EKS.

Considerações

- O driver da CSI do Amazon EFS não é compatível com imagens de contêiner baseadas no Windows.
- Você não pode usar o provisionamento dinâmico de volumes persistentes com nós do Fargate, mas pode usar o provisionamento estático.

- O provisionamento dinâmico exige a versão `1.2` ou posterior do driver. Você pode provisionar estaticamente os volumes persistentes usando a versão `1.1` do driver em qualquer versão compatível do cluster do Amazon EKS.
- A versão `1.3.2` ou superior deste driver é compatível com a arquitetura Arm64, inclusive instâncias baseadas no Graviton do Amazon EC2.
- A versão `1.4.2` ou posterior deste driver suporta o uso de FIPS para montar sistemas de arquivos.
- Anote as cotas de recursos do Amazon EFS. Por exemplo, há uma cota de 1000 pontos de acesso que podem ser criados para cada sistema de arquivos do Amazon EFS. Para obter mais informações, consulte Cotas de recursos do Amazon EFS que não podem ser alteradas.

Pré-requisitos

- Um provedor OpenID Connect (OIDC) do AWS Identity and Access Management (IAM) existente para o cluster. Para determinar se você tem ou para criar uma, consulte Criar um provedor OIDC do IAM para o cluster.
- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.

Etapa 1: Criar uma função do IAM

O driver da CSI do Amazon EFS exige permissões do IAM para interagir com seu sistema de arquivos. Crie um perfil do IAM de cluster e associe a política gerenciada do AWS a ele. É possível usar o `eksctl`, o AWS Management Console ou a AWS CLI.

> **Note**
>
> As etapas específicas desse procedimento foram escritas para usar o driver como um complemento do Amazon EKS. Para obter detalhes sobre instalações autogerenciadas, consulte Set up driver permission no GitHub.

eksctl

Para criar seu perfil do IAM do driver da CSI do Amazon EFS com **`eksctl`**

Execute os comandos a seguir para criar o perfil do IAM. Substitua *`my-cluster`* pelo nome do seu cluster e *`AmazonEKS_EFS_CSI_DriverRole`* pelo nome desejado para seu perfil.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEFSCSIDriverPolicy \
    --approve
TRUST_POLICY=$(aws iam get-role --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

AWS Management Console

Para criar seu perfil do IAM do driver da CSI do Amazon EFS com o AWS Management Console

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.

2. No painel de navegação à esquerda, escolha Roles.
3. Na página Roles (Funções), selecione Create role (Criar função).
4. Na página Select trusted entity (Selecionar entidade confiável), faça o seguinte:
   a. Na seção Trusted entity type (Tipo de entidade confiável), escolha Web identity (Identidade da Web).
   b. Em Identity provider (Provedor de identidade), escolha OpenID ConnectOpenID Connect provider URL (URL do provedor OpenID Connect) para o cluster (conforme mostrado na guia Overview [Visão geral] do Amazon EKS).
   c. Para Audience (Público), escolha `sts.amazonaws.com`.
   d. Escolha Próximo.
5. Na página Add permissions (Adicionar permissões), faça o seguinte:
   a. Na caixa Filter policies (Políticas de filtro) insira *`AmazonEFSCSIDriverPolicy`*.
   b. Marque a caixa de seleção à esquerda do *`AmazonEFSCSIDriverPolicy`* retornado na pesquisa.
   c. Escolha Próximo.
6. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:
   a. Em Role name (Nome da função), insira um nome exclusivo para a função, como ***`AmazonEKS_EFS_CSI_DriverRole`***.
   b. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.
   c. Selecione Criar função.
7. Depois que a função for criada, escolha a função no console a fim de abri-la para edição.
8. Escolha a guia Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).
9. Encontre a linha semelhante à seguinte:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

Adicione a seguinte linha acima da linha anterior. Substitua *region-code* pela Região da AWS em que está o cluster. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* pelo ID do provedor OIDC do cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:efs-csi-*",
```

10. Modifique o operador `Condition` de `"StringEquals"` para `"StringLike"`.
11. Escolha Update Policy (Atualizar política) para concluir.

AWS CLI

Para criar seu perfil do IAM do driver da CSI do Amazon EFS com o AWS CLI

1. Exiba a URL do provedor OIDC do cluster. Substitua *my-cluster* pelo nome do cluster. Se o resultado do comando for `None`, revise os Pré-requisitos.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

Veja um exemplo de saída abaixo.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Crie um perfil do IAM que conceda a ação `AssumeRoleWithWebIdentity`.

   a. Copie o conteúdo a seguir em um arquivo denominado *aws-efs-csi-driver-trust-policy*`.json`. Substitua *111122223333* pelo ID da sua conta. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* e *region-code* pelos valores retornados na etapa anterior. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:efs-csi-*",
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

b. Crie a função. Você pode alterar *AmazonEKS_EFS_CSI_DriverRole* para usar um nome diferente, mas se o fizer, altere-o também nas etapas seguintes.

```
aws iam create-role \
  --role-name AmazonEKS_EFS_CSI_DriverRole \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-
policy.json"
```

3. Anexe a política gerenciada pela AWS necessária à função com o comando a seguir. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name AmazonEKS_EFS_CSI_DriverRole
```

Etapa 2: obter o driver de CSI do Amazon EFS

Recomendamos que você instale o driver da CSI do Amazon EFS por meio do complemento Amazon EKS. Para adicionar um complemento do Amazon EKS ao cluster, consulte Criar um complemento do Amazon EKS. Para obter mais informações sobre complementos, consulte Complementos do Amazon EKS. Se você não conseguir usar o complemento do Amazon EKS,

recomendamos que envie um problema sobre o problema que está impedindo o uso do repositório no GitHub com roteiro de contêineres.

Como alternativa, se quiser uma instalação autogerenciada do driver da CSI do Amazon EFS, consulte Instalação em GitHub.

Etapa 3: criar um sistema de arquivos do Amazon EFS

> **Note**
>
> Essa etapa não é necessária para o AWS Fargate. Um Pod em execução no Fargate monta automaticamente um sistema de arquivos do Amazon EFS.

Para criar um sistema de arquivos do Amazon EFS, consulte Criar um sistema de arquivos do Amazon EFS para o Amazon EKS no GitHub.

Etapa 4: implantar uma aplicação de exemplo

Você pode implantar toda uma variedade de aplicações de exemplo e modificá-las conforme necessário. Para obter mais informações, consulte Exemplos no GitHub.

Armazene aplicações de alta performance com o FSx para Lustre

O driver de Container Storage Interface (CSI) do FSx for Lustre fornece uma interface CSI que permite que os clusters do Amazon EKS gerenciem o ciclo de vida dos sistemas de arquivos do FSx for Lustre. Para obter mais informações, consulte o Guia do usuário do FSx para Lustre.

Este tópico mostra como implantar o driver de CSI do FSx for Lustre no cluster do Amazon EKS e verificar se ele funciona. Recomendamos o uso da versão mais recente do driver. Para ver as versões disponíveis, consulte CSI Specification Compatibility Matrix (Matriz de compatibilidade de especificações de CSI) no GitHub.

> **Note**
>
> O driver não é compatível com o Fargate.

Para obter descrições detalhadas sobre os parâmetros disponíveis e exemplos completos que demonstram os recursos do driver, consulte o projeto FSx for Lustre Container Storage Interface (CSI) driver (Driver de Container Storage Interface (CSI) do FSx for Lustre) no GitHub.

Pré-requisitos

Você deve ter:

- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.
- Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.

Os procedimentos a seguir ajudam a criar um cluster simples de teste com o driver de CSI do FSx for Lustre para que você veja como ele funciona. Não recomendamos utilizar o cluster de teste para workloads de produção. Para este tutorial, convém utilizar os *`example values`*, exceto onde indicado para substituí-los. É possível substituir qualquer *`example value`* ao concluir as etapas para um cluster de produção. Convém concluir todas as etapas no mesmo terminal, pois variáveis são definidas e utilizadas por todas as etapas e não existirão em terminais diferentes.

Para implantar o driver CSI do FSx para Lustre

1. Defina algumas variáveis para uso nas etapas restantes. Substitua *`my-csi-fsx-cluster`* pelo nome do cluster de teste que você deseja criar e *`region-code`* pela Região da AWS na qual você deseja criar o cluster de teste.

```
export cluster_name=my-csi-fsx-cluster
export region_code=region-code
```

2. Crie um cluster de teste.

```
eksctl create cluster \
  --name $cluster_name \
  --region $region_code \
  --with-oidc \
  --ssh-access \
  --ssh-public-key my-key
```

O provisionamento de cluster leva alguns minutos. Durante a criação do cluster, você verá várias linhas de saída. A última linha do resultado é semelhante ao exemplo de linha a seguir.

```
[#]  EKS cluster "my-csi-fsx-cluster" in "region-code" region is ready
```

3. Crie uma conta de serviço do Kubernetes para o driver e anexe a política gerenciada pela AWS `AmazonFSxFullAccess` à conta de serviço. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
eksctl create iamserviceaccount \
  --name fsx-csi-controller-sa \
  --namespace kube-system \
  --cluster $cluster_name \
  --attach-policy-arn arn:aws:iam::aws:policy/AmazonFSxFullAccess \
  --approve \
  --role-name AmazonEKSFSxLustreCSIDriverFullAccess \
  --region $region_code
```

Você verá várias linhas de resultado à medida que a conta de serviço é criada. As últimas linhas de saída são semelhantes às seguintes:

```
[#]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/fsx-csi-controller-sa",
        create serviceaccount "kube-system/fsx-csi-controller-sa",
    } }
```

```
[#]  building iamserviceaccount stack "eksctl-my-csi-fsx-cluster-addon-
iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#]  deploying stack "eksctl-my-csi-fsx-cluster-addon-iamserviceaccount-kube-
system-fsx-csi-controller-sa"
[#]  waiting for CloudFormation stack "eksctl-my-csi-fsx-cluster-addon-
iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#]  created serviceaccount "kube-system/fsx-csi-controller-sa"
```

Observe o nome da pilha do AWS CloudFormation implantada. No resultado do exemplo anterior, a pilha é denominada `eksctl-my-csi-fsx-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa`.

4. Implante o driver com o seguinte comando. Substitua *release-X.XX* pela ramificação desejada. A ramificação principal não é aceita porque ela pode conter recursos futuros incompatíveis com a versão estável do driver lançada. Recomendamos o uso da versão mais recente. Para ver uma lista de ramificações, consulte Ramificações do `aws-fsx-csi-driver` no GitHub.

> **Note**
>
> É possível visualizar o conteúdo que está sendo aplicado em `aws-fsx-csi-driver/deploy/kubernetes/overlays/stable` no GitHub.

```
kubectl apply -k "github.com/kubernetes-sigs/aws-fsx-csi-driver/deploy/kubernetes/
overlays/stable/?ref=release-X.XX"
```

Veja um exemplo de saída abaixo.

```
serviceaccount/fsx-csi-controller-sa created
serviceaccount/fsx-csi-node-sa created
clusterrole.rbac.authorization.k8s.io/fsx-csi-external-provisioner-role created
clusterrole.rbac.authorization.k8s.io/fsx-external-resizer-role created
clusterrolebinding.rbac.authorization.k8s.io/fsx-csi-external-provisioner-binding
 created
clusterrolebinding.rbac.authorization.k8s.io/fsx-csi-resizer-binding created
deployment.apps/fsx-csi-controller created
daemonset.apps/fsx-csi-node created
csidriver.storage.k8s.io/fsx.csi.aws.com created
```

5. Observe o ARN para a função que ela foi criada. Se você não o anotou antes e ele não o estiver mais disponível na saída da AWS CLI, faça o seguinte para visualizá-lo no AWS Management Console:

   a. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.

   b. Verifique se o console está definido para a Região da AWS na qual você criou a função do IAM e selecione Stacks (Pilhas).

   c. Selecione a pilha chamada `eksctl-`*`my-csi-fsx-cluster`*`-addon-iamserviceaccount-kube-system-`*`fsx-csi-controller-sa`*.

   d. Selecione a guia Outputs (Resultados). O ARN de Role1 é listado na página Outputs (1) (Saídas).

6. Aplique um patch à implantação do driver para adicionar a conta de serviço criada anteriormente com o comando a seguir. Substitua o ARN pelo ARN que você anotou. Substitua *`111122223333`* pelo ID da sua conta. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

   ```
   kubectl annotate serviceaccount -n kube-system fsx-csi-controller-sa \
     eks.amazonaws.com/role-
   arn=arn:aws:iam::111122223333:role/AmazonEKSFSxLustreCSIDriverFullAccess --
   overwrite=true
   ```

   Veja um exemplo de saída abaixo.

   ```
   serviceaccount/fsx-csi-controller-sa annotated
   ```

Para implantar uma classe de armazenamento, uma solicitação de volume persistente e uma aplicação de amostra

Este procedimento usa o repositório FSx for Lustre Container Storage Interface (CSI) driver (Driver de Container Storage Interface (CSI) do FSx for Lustre) para consumir um volume provisionado dinamicamente do FSx for Lustre) do GitHub.

1. Observe o grupo de segurança do seu cluster. É possível ver isso noAWS Management Console, na seção Networking (Redes), ou usando o seguinte comando da AWS CLI:

```
aws eks describe-cluster --name $cluster_name --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

2. Crie um grupo de segurança para seu sistema de arquivos Amazon FSx conforme os critérios mostrados em Grupos de segurança de Amazon VPC, no Guia do usuário do Amazon FSx para Lustre. Para a VPC, selecione a VPC do seu cluster, conforme mostrado na seção Networking (Redes). Para os “grupos de segurança associados aos seus clientes Lustre”, utilize o grupo de segurança do seu cluster. É possível deixar as regras de saída sozinhas para permitir Todo o tráfego.
3. Baixe o manifesto da classe de armazenamento com o seguinte comando:

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/
master/examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
```

4. Edite a seção de parâmetros do arquivo `storageclass.yaml`. Substitua *`example value`* por seus próprios valores.

```
parameters:
  subnetId: subnet-0eabfaa81fb22bcaf
  securityGroupIds: sg-068000ccf82dfba88
  deploymentType: PERSISTENT_1
  automaticBackupRetentionDays: "1"
  dailyAutomaticBackupStartTime: "00:00"
  copyTagsToBackups: "true"
  perUnitStorageThroughput: "200"
  dataCompressionType: "NONE"
  weeklyMaintenanceStartTime: "7:09:00"
  fileSystemTypeVersion: "2.12"
```

- **`subnetId`**: o ID de da sub-rede em que o sistema de arquivos do Amazon FSx para Lustre deve ser criado. O Amazon FSx for Lustre não conta com suporte em todas as zonas de disponibilidade. Abra o console do Amazon FSx for Lustre em https://console.aws.amazon.com/fsx/ para confirmar se a sub-rede que você deseja usar está em uma zona de disponibilidade compatível. A sub-rede pode incluir os nós ou pode ser uma sub-rede ou uma VPC diferente:
  - É possível verificar as sub-redes do nó no AWS Management Console, selecionando o grupo de nós na seção Compute (Computação).

- Se a sub-rede especificada não for a mesma sub-rede em que os nós estiverem localizados, as VPCs deverão estar conectadas, e você deverá garantir que as portas necessárias estejam abertas nos grupos de segurança.

- **`securityGroupIds`**: o ID do grupo de segurança criado para o sistema de arquivos.
- **`deploymentType`** (opcional): o tipo de implantação do sistema de arquivos. Os valores válidos são `SCRATCH_1`, `SCRATCH_2`, `PERSISTENT_1` e `PERSISTENT_2`. Para obter mais informações sobre os tipos de implantação, consulte Create your Amazon FSx for Lustre file system (Criar o sistema de arquivos do Amazon FSx for Lustre).
- outros parâmetros (opcionais): para obter informações sobre outros parâmetros, consulte Editar StorageClass (Editar StorageClass) no GitHub.

5. Crie o manifesto da classe de armazenamento.

```
kubectl apply -f storageclass.yaml
```

Veja um exemplo de saída abaixo.

```
storageclass.storage.k8s.io/fsx-sc created
```

6. Faça download do manifesto de declaração de volume persistente.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/
master/examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. (Opcional) Edite o arquivo `claim.yaml`. Altere *`1200Gi`* para um dos seguintes valores de incremento com base nos requisitos de armazenamento e no `deploymentType` selecionado em uma etapa anterior.

```
storage: 1200Gi
```

- SCRATCH_2 e PERSISTENT a **`1.2 TiB`**, **`2.4 TiB`** ou incrementos de 2,4 TiB, com mais de 2,4 TiB.
- SCRATCH_1 a **`1.2 TiB`**, **`2.4 TiB`**, **`3.6 TiB`** ou incrementos de 3,6 TiB, com mais de 3,6 TiB.

8. Crie a declaração de volume persistente.

```
kubectl apply -f claim.yaml
```

Veja um exemplo de saída abaixo.

```
persistentvolumeclaim/fsx-claim created
```

9. Confirme se o sistema de arquivos está provisionado.

```
kubectl describe pvc
```

Veja um exemplo de saída abaixo.

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
[...]
```

> **Note**
>
> O `Status` pode ser exibido como `Pending` por 5 a 10 minutos, antes de mudar para `Bound`. Não prossiga para a próxima etapa até o `Status` for `Bound`. Se `Status` mostrar `Pending` por mais de 10 minutos, use mensagens de aviso em `Events` como referência para resolver problemas.

10. Implante a aplicação de exemplo.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11. Verifique se a aplicação de exemplo está em execução.

```
kubectl get pods
```

Veja um exemplo de saída abaixo.

```
NAME      READY   STATUS              RESTARTS   AGE
fsx-app   1/1     Running             0          8s
```

12. Verifique se o sistema de arquivos está montado corretamente pela aplicação.

```
kubectl exec -ti fsx-app -- df -h
```

Veja um exemplo de saída abaixo.

```
Filesystem                 Size  Used Avail Use% Mounted on
overlay                     80G  4.0G   77G   5% /
tmpfs                       64M     0   64M   0% /dev
tmpfs                      3.8G     0  3.8G   0% /sys/fs/cgroup
192.0.2.0@tcp:/abcdef01    1.1T  7.8M  1.1T   1% /data
/dev/nvme0n1p1              80G  4.0G   77G   5% /etc/hosts
shm                         64M     0   64M   0% /dev/shm
tmpfs                      6.9G   12K  6.9G   1% /run/secrets/kubernetes.io/
serviceaccount
tmpfs                      3.8G     0  3.8G   0% /proc/acpi
tmpfs                      3.8G     0  3.8G   0% /sys/firmware
```

13. Verifique se os dados foram gravados no sistema de arquivos do FSx for Lustre pela aplicação de exemplo.

```
kubectl exec -it fsx-app -- ls /data
```

Veja um exemplo de saída abaixo.

```
out.txt
```

Essa saída de exemplo indica que a aplicação de amostra gravou com êxito o arquivo `out.txt` no sistema de arquivos.

> **Note**
>
> Antes de excluir o cluster, não esqueça de excluir o sistema de arquivos do FSx for Lustre. Para obter mais informações, consulte Limpar os recursos no Guia do usuário do FSx for Lustre.

Armazene aplicações de alta performance com o FSx para NetApp ONTAP

O Astra Trident da NetApp's fornece orquestração de armazenamento dinâmico utilizando um driver que é compatível com a CSI (Container Storage Interface). Isso permite que clusters do Amazon EKS gerenciem o ciclo de vida dos volumes persistentes (PVs) com suporte por sistemas de arquivo Amazon FSx para NetApp ONTAP. Para começar, consulte Usar o Astra Trident com o Amazon FSx para NetApp ONTAP na documentação do Astra Trident.

O Amazon FSx para NetApp ONTAP é um serviço de armazenamento que permite iniciar e executar sistemas de arquivos ONTAP totalmente gerenciados na nuvem. ONTAP é a tecnologia de sistema de arquivos da NetApp's que fornece um conjunto com todos os recursos de acesso a dados e gerenciamento de dados. O FSx para ONTAP fornece os recursos, a performance e as APIs dos sistemas de arquivos on-premises da NetApp com a agilidade, a escalabilidade e a simplicidade dos serviços totalmente gerenciados da AWS. Para obter mais informações, consulte o Guia do usuário do FSx for ONTAP.

Armazenar dados usando o Amazon FSx para OpenZFS

O Amazon FSx para OpenZFS é um serviço de armazenamento de arquivos totalmente gerenciado que facilita a transferência de dados do ZFS on-premises ou de outros servidores de arquivos baseados em Linux para a AWS. É possível fazer isso sem alterar o código da aplicação ou a forma como seus dados são gerenciados. Ele oferece armazenamento de arquivos altamente confiável, escalável, eficiente e rico em recursos, construído no sistema de arquivos de código aberto OpenZFS. Ele combina esses recursos com a agilidade, a escalabilidade e a simplicidade de um serviço da AWS totalmente gerenciado. Para obter mais informações, consulte o Guia do usuário do Amazon FSx para OpenZFS.

O driver Container Storage Interface (CSI) do FSx para OpenZFS fornece uma interface de CSI que permite que os clusters do Amazon EKS gerenciem o ciclo de vida de volumes do FSx para OpenZFS. Para implantar o driver CSI do FSx para OpenZFS no cluster do Amazon EKS, consulte `aws-fsx-openzfs-csi-driver` no GitHub.

Minimizar a latência com o Amazon File Cache

O Amazon File Cache é um cache de alta velocidade totalmente gerenciado na AWS, que é usado para processar dados de arquivos, independentemente do local em que os dados estão

armazenados. O Amazon File Cache carrega os dados no cache automaticamente quando é acessado pela primeira vez e libera os dados quando não é usado. Para obter mais informações, consulte o Guia do usuário do Amazon File Cache.

O driver da interface de armazenamento de contêiner (CSI) do Amazon File Cache fornece uma interface CSI que possibilita que os clusters do Amazon EKS gerenciem o ciclo de vida dos caches de arquivos da Amazon. Para implantar o driver da CSI do Amazon File Cache em seu cluster do Amazon EKS, consulte `aws-file-cache-csi-driver` no GitHub.

Armazenar dados com a interface web do Amazon S3

Com o Mountpoint para o driver Container Storage Interface (CSI) do Amazon S3, as aplicações do Kubernetes podem acessar objetos do Amazon S3 por meio de uma interface de sistema de arquivos, alcançando alto throughput agregado sem alterar nenhum código da aplicação. Desenvolvido com base no Mountpoint para Amazon S3, o driver da CSI apresenta um bucket do Amazon S3 como um volume que pode ser acessado por contêineres no Amazon EKS e em clusters do Kubernetes autogerenciados. Este tópico mostra como implantar o Mountpoint para driver da CSI do Amazon S3 no cluster do Amazon EKS.

Considerações

- O Mountpoint para driver da CSI do Amazon S3 não é compatível com imagens de contêiner baseadas no Windows.
- O Mountpoint para driver da CSI do Amazon S3 não é compatível com o AWS Fargate. No entanto, os contêineres que executados no Amazon EC2 (com o Amazon EKS ou com uma instalação personalizada do Kubernetes) são compatíveis.
- O Mountpoint para driver da CSI do Amazon S3 é compatível somente com provisionamento estático. Não há suporte ao provisionamento dinâmico ou à criação de novos buckets.

  > **Note**
  >
  > O provisionamento estático refere-se ao uso de um bucket do Amazon S3 existente que é especificado como o `bucketName` no `volumeAttributes` no objeto `PersistentVolume`. Para obter mais informações, consulte Provisionamento estático no GitHub.

- Os volumes montados com o Mountpoint para driver da CSI do Amazon S3 não oferecem suporte a todos os recursos do sistema de arquivos POSIX. Para obter detalhes sobre o comportamento

do sistema de arquivos, consulte Comportamento do Mountpointpara sistema de arquivos do Amazon S3 no GitHub.

Pré-requisitos

- Um provedor OpenID Connect (OIDC) do AWS Identity and Access Management (IAM) existente para o cluster. Para determinar se você tem ou para criar uma, consulte Criar um provedor OIDC do IAM para o cluster.
- A versão 2.12.3 ou posterior da AWS CLI instalada e configurada em seu dispositivo ou AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.

Criar uma política do IAM

O Mountpoint para driver da CSI do Amazon S3 exige permissões do Amazon S3 para interagir com seu sistema de arquivos. Esta seção mostra como criar uma política do IAM que conceda as permissões necessárias.

O exemplo de política a seguir segue as recomendações de permissão do IAM para Mountpoint. Como alternativa, você pode usar a política AmazonS3FullAccess gerenciada pela AWS, mas essa política gerenciada concede mais permissões do que as necessárias para o Mountpoint.

Para obter mais informações sobre as permissões recomendadas para o Mountpoint, consulte Permissões do IAM do Mountpoint no GitHub.

Para criar uma política do IAM com o console do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Políticas.
3. Na página Políticas, escolha Criar política.
4. Em Editor de políticas, escolha JSON.
5. No Editor de políticas, copie e cole o seguinte:

> **Important**
>
> Substitua `amzn-s3-demo-bucket1` pelo nome do seu próprio bucket do Amazon S3.

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Sid": "MountpointFullBucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Sid": "MountpointFullObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
        }
   ]
}
```

Os buckets de diretório, introduzidos com a classe de armazenamento Amazon S3 Express One Zone, usam um mecanismo de autenticação diferente dos buckets de uso geral. Em vez de usar ações `s3:*`, você deve usar a ação `s3express:CreateSession`. Para obter mais informações sobre os buckets do diretório, consulte Buckets de diretório no Guia do usuário do Amazon S3.

Abaixo está um exemplo de política de privilégio mínimo que você usaria para um bucket de diretório.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "arn:aws:s3express:aws-region:111122223333:bucket/amzn-s3-demo-bucket1--az_id--x-s3"
        }
    ]
}
```

6. Escolha Próximo.
7. Na página Analisar, atribua um nome à sua política. Esta demonstração usa o nome `AmazonS3CSIDriverPolicy`.
8. Escolha Criar política.

Criar um perfil do IAM

O Mountpoint para driver da CSI do Amazon S3 exige permissões do Amazon S3 para interagir com seu sistema de arquivos. Esta seção mostra como criar um perfil do IAM para delegar essas permissões. Para criar esse perfil, é possível usar o `eksctl`, o console do IAM ou a AWS CLI.

> **Note**
>
> A política do IAM `AmazonS3CSIDriverPolicy` foi criada na seção anterior.

eksctl

Para criar seu perfil do IAM do Mountpoint para o driver da CSI do Amazon S3 com o **`eksctl`**

Para criar o perfil do IAM e a conta de serviço do Kubernetes, execute os comandos a seguir. Esses comandos também anexam a política do IAM `AmazonS3CSIDriverPolicy` ao perfil, anotam a conta de serviço do Kubernetes (`s3-csi-controller-sa`) com o nome do recurso

da Amazon (ARN) do perfil do IAM e adicionam o nome da conta de serviço do Kubernetes à política de confiança para o perfil do IAM.

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

IAM console

Para criar o perfil do IAM do Mountpoint para driver da CSI do Amazon S3 com o AWS Management Console

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.
3. Na página Roles (Funções), selecione Create role (Criar função).
4. Na página Select trusted entity (Selecionar entidade confiável), faça o seguinte:
   a. Na seção Trusted entity type (Tipo de entidade confiável), escolha Web identity (Identidade da Web).
   b. Em Identity provider (Provedor de identidade), escolha OpenID ConnectOpenID Connect provider URL (URL do provedor OpenID Connect) para o cluster (conforme mostrado na guia Overview [Visão geral] do Amazon EKS).

      Se nenhum URL for mostrado, revise a seção Pré-requisitos.
   c. Para Audience (Público), escolha `sts.amazonaws.com`.
   d. Escolha Próximo.
5. Na página Add permissions (Adicionar permissões), faça o seguinte:
   a. Na caixa Filter policies (Políticas de filtro) insira **`AmazonS3CSIDriverPolicy`**.

> **Note**
>
> Essa política foi criada na seção anterior.

b. Marque a caixa de seleção à esquerda do resultado `AmazonS3CSIDriverPolicy` retornado na pesquisa.

c. Escolha Próximo.

6. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:

a. Em Role name (Nome da função), insira um nome exclusivo para a função, como **`AmazonEKS_S3_CSI_DriverRole`**.

b. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.

c. Selecione Criar função.

7. Depois que a função for criada, escolha a função no console a fim de abri-la para edição.

8. Escolha a guia Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).

9. Encontre a linha semelhante à seguinte:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

Adicione uma vírgula ao final da linha anterior e, em seguida, adicione a linha a seguir após ela. Substitua *`region-code`* pela Região da AWS em que está o cluster. Substitua *`EXAMPLED539D4633E53DE1B71EXAMPLE`* pelo ID do provedor OIDC do cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:s3-csi-*"
```

10. Altere o operador `Condition` de `"StringEquals"` para `"StringLike"`.

11. Escolha Update Policy (Atualizar política) para concluir.

AWS CLI

Para criar o perfil do IAM do Mountpoint para driver da CSI do Amazon S3 com o AWS CLI

1. Visualize o URL do provedor OIDC do cluster. Substitua o *my-cluster* pelo nome do cluster. Se o resultado do comando for None, revise os Pré-requisitos.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

Veja um exemplo de saída abaixo.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Crie o perfil do IAM, concedendo à conta de serviço do Kubernetes a ação AssumeRoleWithWebIdentity.

a. Copie o conteúdo a seguir em um arquivo denominado *aws-s3-csi-driver-trust-policy*.json. Substitua *111122223333* pelo ID da sua conta. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* e *region-code* pelos valores retornados na etapa anterior.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:s3-csi-*",
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
```

```
    }
  ]
}
```

b. Crie a função. Você pode alterar *AmazonEKS_S3_CSI_DriverRole* para usar um nome diferente, mas se o fizer, altere-o também nas etapas seguintes.

```
aws iam create-role \
  --role-name AmazonEKS_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"
```

3. Anexe a política do IAM criada anteriormente ao perfil executando o comando a seguir.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole
```

> **Note**
>
> A política do IAM `AmazonS3CSIDriverPolicy` foi criada na seção anterior.

4. Pule esta etapa se você estiver instalando o driver como um complemento do Amazon EKS.. Para instalações autogerenciadas do driver, crie contas de serviço do Kubernetes que detalhem o ARN do perfil do IAM que você criou.

a. Salve o conteúdo a seguir em um arquivo denominado *mountpoint-s3-service-account*`.yaml`. Substitua *111122223333* pelo ID da sua conta.

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: mountpoint-s3-csi-controller-sa
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn:
 arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
```

b. Crie a conta de serviço Kubernetes no cluster: A conta de serviço do Kubernetes (`mountpoint-s3-csi-controller-sa`) é anotada com o perfil do IAM criado com o nome *AmazonEKS_S3_CSI_DriverRole*.

```
kubectl apply -f mountpoint-s3-service-account.yaml
```

> **Note**
>
> Quando você implanta o plug-in nesse procedimento, ele cria e é configurado para usar uma conta de serviço chamada `s3-csi-driver-sa`.

Instalar o Mountpoint para o driver CSI do Amazon S3

É possível instalar o Mountpoint para driver da CSI do Amazon S3 via complemento Amazon EKS. É possível usar `eksctl`, o AWS Management Console ou a AWS CLI para adicionar o complemento ao seu cluster.

Você também pode instalar o driver CSI Mountpoint do Amazon S3 como uma instalação autogerenciada. Para obter instruções sobre como realizar uma instalação autogerenciada, consulte Instalação em GitHub.

eksctl

Para adicionar o complemento CSI do Amazon S3 usando **`eksctl`**

Execute o seguinte comando . Substitua *my-cluster* pelo nome do cluster, *111122223333* pelo ID da conta e *AmazonEKS_S3_CSI_DriverRole* pelo nome do perfil do IAM criado anteriormente.

```
eksctl create addon --name aws-mountpoint-s3-csi-driver --cluster my-cluster --
service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole --force
```

Se você remover a opção ***--force*** e qualquer uma das configurações de complemento do Amazon EKS entra em conflito com suas configurações existentes, então a atualização do complemento Amazon EKS falha e você recebe uma mensagem de erro para ajudá-lo a resolver o conflito. Antes de especificar essa opção, certifique-se de que o complemento Amazon

EKS não gerencie as configurações que você precisa gerenciar, pois essas configurações são substituídas por essa opção. Para obter mais informações sobre outras opções para essa configuração, consulte Addons (Complementos) na documentação do `eksctl`. Para obter mais informações sobre o gerenciamento de campos do Amazon EKS Kubernetes, consulte Determinar os campos que podem ser personalizados para os complementos do Amazon EKS.

AWS Management Console

Para adicionar o Mountpoint para o complemento CSI do Amazon S3 usando o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome do cluster para o qual você deseja configurar o Mountpoint para o complemento CSI do Amazon S3.
4. Escolha a guia Add-ons (Complementos).
5. Escolha Obter mais complementos.
6. Na página Selecionar complementos, faça o seguinte:
    a. Na seção Complementos do Amazon EKS, marque a caixa de seleção Mountpoint para o driver de CSI do Amazon S3.
    b. Escolha Próximo.
7. Na página Definir as configurações dos complementos selecionados:
    a. Selecione a Versão que deseja usar.
    b. Em Selecionar perfil do IAM, selecione o nome do perfil do IAM ao qual você anexou o Mountpoint para a política do IAM do driver de CSI do Amazon S3.
    c. (Opcional) Você pode expandir Definições de configuração opcionais. Se você selecionar Substituir como Método de resolução de conflitos, uma ou mais configurações do complemento existente poderão ser substituídas pelas configurações do complemento do Amazon EKS. Se você não habilitar esta opção e houver um conflito com suas configurações existentes, a operação falhará. É possível usar a mensagem de erro resultante para solucionar o conflito. Antes de selecionar essa opção, certifique-se de que o complemento do Amazon EKS não gerencie as configurações que você precisa autogerenciar.
    d. Escolha Próximo.

8. Na página Adicionar tags, escolha Criar. Depois que a instalação do complemento for concluída, você verá o complemento instalado.

AWS CLI

Para adicionar o Mountpoint para o complemento CSI do Amazon S3 usando o AWS CLI

Execute o seguinte comando . Substitua *my-cluster* pelo nome do cluster, *111122223333* pelo ID da conta e *AmazonEKS_S3_CSI_DriverRole* pelo nome da função criada anteriormente.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
```

Configurar o Mountpoint para o Amazon S3

Na maioria dos casos, você pode configurar o Mountpoint para o Amazon S3 com apenas um nome de bucket. Para obter instruções sobre como configurar o Mountpoint para Amazon S3, consulte Configurar o Mountpoint para Amazon S3 no GitHub.

Implantar uma aplicação de exemplo

É possível implantar provisionamento estático para o driver em um bucket do Amazon S3 existente. Para obter mais informações, consulte Provisionamento estático no GitHub.

Remover o Mountpoint do driver CSI do Amazon S3

Você tem duas opções ao remover um complemento do Amazon EKS.

- Preserve add-on software on your cluster (Preservar software de complemento no cluster): esta opção remove o gerenciamento do Amazon EKS de qualquer configuração. Também remove a capacidade do Amazon EKS de notificar você sobre atualizações e de atualizar automaticamente o complemento do Amazon EKS depois de iniciar uma atualização. No entanto, ele preserva o software de complemento em seu cluster. Essa opção torna o complemento em uma instalação autogerenciada, em vez de um complemento do Amazon EKS. Com essa opção, não há tempo de inatividade para o complemento. Os comandos deste procedimento usam essa opção.

- Remover completamente o software do complemento do cluster: recomendamos remover o complemento do Amazon EKS do cluster se não houver recursos no cluster que dependam dele. Para fazer essa opção, exclua `--preserve` a partir do comando que você utiliza neste procedimento.

Se não houver uma conta do IAM associada ao complemento, a conta do IAM não será removida.

É possível usar `eksctl`, o AWS Management Console ou a AWS CLI para remover o complemento CSI do Amazon S3.

eksctl

Para remover o complemento CSI do Amazon S3 usando **`eksctl`**

Substitua *`my-cluster`* pelo nome do cluster e execute comando a seguir.

```
eksctl delete addon --cluster my-cluster --name aws-mountpoint-s3-csi-driver --
preserve
```

AWS Management Console

Para remover o complemento CSI do Amazon S3 usando o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome do cluster de onde você deseja remover o complemento CSI do Amazon EBS.
4. Escolha a guia Add-ons (Complementos).
5. Escolha Mountpoint para driver de CSI do Amazon S3.
6. Escolha Remover.
7. Na caixa de diálogo de confirmação Remover: aws-mountpoint-s3-csi-driver, faça o seguinte:
   a. Se quiser que o Amazon EKS pare de gerenciar as configurações do complemento, selecione Preservar no cluster. Faça isso se quiser manter o software de complemento em seu cluster. Isso serve para que você possa gerenciar todas as configurações do complemento por conta própria.
   b. Digite **`aws-mountpoint-s3-csi-driver`**.

c. Selecione Remover.

AWS CLI

Para remover o complemento CSI do Amazon S3 usando o AWS CLI

Substitua *my-cluster* pelo nome do cluster e execute comando a seguir.

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-driver --preserve
```

Habilitar a funcionalidade de snapshot para volumes CSI

Quando uma Container Storage Interface (CSI) é usada, a funcionalidade de snapshots permite criar cópias pontuais de seus dados. Para que esse recurso funcione no Kubernetes, um driver de CSI com suporte a snapshots (como o driver CSI do Amazon EBS) e um controlador de snapshots CSI são necessários. O controlador de snapshot está disponível como complemento gerenciado do Amazon EKS ou como uma instalação autogerenciada.

Veja a seguir algumas considerações ao usar o controlador de snapshots da CSI.

- O controlador de snapshots deve ser instalado junto com um driver de CSI com funcionalidade de snapshot instantâneo. Por exemplo, o driver de CSI do Amazon EBS implementa a captura de instantâneos como um contêiner separado chamado. `csi-snapshotter` Este contêiner pode criar snapshots do Amazon EBS de volumes gerenciados pela CSI do Amazon EBS. Para obter instruções sobre como instalar o driver de CSI do Amazon EBS no cluster, consulte Armazene volumes do Kubernetes com o Amazon EBS.
- O Kubernetes não oferece suporte a snapshots de volumes que estão sendo servidos por meio da migração da CSI, como volumes do Amazon EBS que usam uma `StorageClass` com provisionador do `kubernetes.io/aws-ebs`. Os volumes devem ser criados com um `StorageClass` que faça referência ao provisionador do driver da CSI, `ebs.csi.aws.com`. Para obter mais informações sobre a migração da CSI, consulte Perguntas frequentes sobre migração de CSI do Amazon EBS.

Recomendamos instalar o controlador de snapshots da CSI via complemento gerenciado do Amazon EKS. Para adicionar um complemento do Amazon EKS ao cluster, consulte Criar um complemento

do Amazon EKS. Para obter mais informações sobre complementos, consulte Complementos do Amazon EKS.

Como alternativa, se você quiser uma instalação autogerenciada do controlador de snapshots da CSI do Amazon EBS, consulte Uso no `external-snapshotter` do Kubernetes upstream no GitHub

Rede Amazon EKS

Seu cluster do Amazon EKS é criado em uma VPC. A rede de pod é fornecida pelo plugin Container Network Interface (CNI) da Amazon VPC. Este capítulo inclui os tópicos a seguir para aprender mais sobre redes para o cluster.

Tópicos

Requisitos e considerações sobre a VPC e a sub-rede do Amazon EKS

Ao criar um cluster, você especifica uma VPC e pelo menos duas sub-redes em diferentes zonas de disponibilidade. Este tópico fornece uma visão geral dos requisitos e das considerações específicos do Amazon EKS para a VPC e as sub-redes que você utiliza com o cluster. Se você não tiver uma VPC para utilizar com o Amazon EKS, poderá criar uma utilizando um modelo AWS CloudFormation fornecido pelo Amazon EKS. Se você estiver criando um cluster local ou estendido no AWS Outposts, consulte Criar uma VPC e sub-redes para clusters do Amazon EKS no AWS Outposts em vez deste tópico.

Requisitos e considerações para VPCs

Quando você cria um cluster, a VPC especificada deve atender aos requisitos e considerações a seguir:

- A VPC deverá ter uma quantidade suficiente de endereços IP disponíveis para o cluster, todos os nós e outros recursos do Kubernetes que você desejar criar. Se a VPC que você deseja utilizar não tiver uma quantidade suficiente de endereços IP, tente aumentar a quantidade de endereços IP disponíveis.

  Você pode fazer isso atualizando a configuração do cluster para alterar as sub-redes e grupos de segurança que o cluster utiliza. Você pode atualizar AWS Management Console a partir da versão

mais recente da versão AWS CLI, AWS CloudFormation e `eksctl` `v0.164.0-rc.0` ou posterior. Talvez seja necessário fazer isso para fornecer às sub-redes mais endereços IP disponíveis para atualizar com sucesso uma versão de cluster.

> **Important**
>
> Todas as sub-redes adicionadas devem estar no mesmo conjunto de AZs fornecido na ocasião da criação do cluster. As novas sub-redes devem atender a todos os outros requisitos, por exemplo, devem ter endereços IP suficientes.
> Por exemplo, suponha que você tenha feito um cluster e especificado quatro sub-redes. Na ordem em que você as especificou, a primeira sub-rede está na Zona de disponibilidade `us-west-2a`, a segunda e a terceira sub-redes estão na Zona de disponibilidade `us-west-2b` e a quarta sub-rede está na Zona de disponibilidade `us-west-2c`. Se quiser alterar as sub-redes, você deverá fornecer pelo menos uma sub-rede em cada uma das três zonas de disponibilidade, e as sub-redes deverão estar na mesma VPC que as sub-redes originais.

Se você precisar de mais endereços IP do que os blocos CIDR na VPC, você pode adicionar mais blocos CIDR associando blocos adicionais de Encaminhamento Entre Domínios Sem Classificação (CIDR) à sua VPC. É possível associar blocos CIDR privados (RFC 1918) e públicos (não RFC 1918) à sua VPC antes ou depois de criar o cluster. Um cluster pode precisar de até cinco horas para que um bloco CIDR associado a uma VPC seja reconhecido.

Você pode conservar a utilização do endereço IP usando um gateway de trânsito com uma VPC de serviços compartilhados. Para obter mais informações, consulte Isolated VPCs with shared services (VPCs isoladas com serviços compartilhados) e Amazon EKS VPC routable IP address conservation patterns in a hybrid network (Padrões de conservação de endereço IP roteáveis da VPC no Amazon EKS em uma rede híbrida).

- Se quiser que o Kubernetes atribua endereços `IPv6` a Pods e serviços, associe um bloco CIDR `IPv6` à VPC. Para obter mais informações, consulte Associar um bloco CIDR `IPv6` à sua VPC, no Guia do usuário da Amazon VPC.
- A VPC deve ter suporte de nomes de host de `DNS` e resolução de `DNS`. Caso contrário, os nós não poderão se registrar no seu cluster. Para mais informações, consulte Atributos de DNS para sua VPC, no Guia do usuário da Amazon VPC.

- A VPC pode exigir endpoints da VPC usando o AWS PrivateLink. Para obter mais informações, consulte Requisitos e considerações para sub-redes.

Se você criou um cluster com o Kubernetes `1.14` ou anterior, o Amazon EKS adicionou a seguinte tag à VPC:

Chave	Valor
`kubernetes.io/cluster/` *`my-cluster`*	`owned`

Essa etiqueta foi utilizada apenas pelo Amazon EKS. É possível remover a etiqueta sem afetar os serviços. Ela não é utilizada com clusters que são da versão `1.15` ou posterior.

Requisitos e considerações para sub-redes

Quando você cria um cluster, o Amazon EKS cria de 2 a 4 interfaces de rede elástica nas sub-redes especificadas. Essas interfaces permitem a comunicação entre seu cluster e sua VPC. Essa interfaces de redes também habilitam recursos do Kubernetes, como `kubectl exec` e `kubectl logs`. Cada interface de rede criada pelo Amazon EKS tem o texto `Amazon EKS` *`cluster-name`* em sua descrição.

O Amazon EKS é capaz de criar suas interfaces de rede em qualquer sub-rede especificada ao criar um cluster. É possível alterar em quais sub-redes o Amazon EKS cria suas interfaces de rede depois que o cluster é criado. Quando você atualiza a versão do Kubernetes de um cluster, o Amazon EKS exclui as interfaces de rede originais ele criou e cria novas interfaces de rede. Essas interfaces podem ser criadas nas mesmas sub-redes que a das interfaces de rede originais ou em sub-redes diferentes das originais. Para controlar quais interfaces de rede de sub-redes são criadas, você poderá limitar o número de sub-redes especificadas a apenas duas ao criar um cluster ou atualizar as sub-redes depois de criar o cluster.

Requisitos de sub-redes para clusters

As sub-redes que você especifica ao criar ou atualizar um cluster devem atender aos requisitos a seguir:

- As sub-redes devem ter no mínimo seis endereços IP para uso pelo Amazon EKS. Porém, recomendamos no mínimo 16 endereços IP.

- As sub-redes não podem residir no AWS Outposts, no AWS Wavelength ou em uma zona local da AWS. Porém, se você as tiver na VPC, poderá implantar nós autogerenciados e recursos do Kubernetes nesses tipos de sub-redes.
- As sub-redes podem ser do tipo público ou privado. Convém, se possível, especificar sub-redes privadas. Uma sub-rede pública tem uma tabela de rotas que inclui uma rota para um gateway da Internet, enquanto uma sub-rede privada tem uma tabela de rotas que não inclui uma rota para um gateway da Internet.
- As sub-redes não podem residir nas seguintes zonas de disponibilidade:

Região da AWS	Nome da região	IDs de zona de disponibilidade não permitidas
`us-east-1`	Leste dos EUA (N. da Virgínia)	`use1-az3`
`us-west-1`	Oeste dos EUA (N. da Califórnia)	`usw1-az2`
`ca-central-1`	Canadá (Central)	`cac1-az3`

Uso da família de endereços IP por componente

A tabela a seguir contém a família de endereços IP usada por cada componente do Amazon EKS. É possível usar uma conversão de endereços de rede (NAT) ou outros sistemas de compatibilidade para efetuar uma conexão com esses componentes usando endereços IP de origem em famílias com o valor "No" para uma entrada de tabela.

A funcionalidade pode diferir dependendo da configuração IP family (`ipFamily`) do cluster. Essa configuração altera o tipo de endereço IP usado para o bloco de CIDR que o Kubernetes atribui para Services. Um cluster com o valor de configuração IPv4 é referido como um IPv4 cluster, e um cluster com o valor de configuração IPv6 é referido como um IPv6 cluster.

Componente	Somente endereços **IPv4**	Somente endereços **IPv6**	Endereços de pilha dupla
Endpoint público da API EKS	Sim[13]	Sim[13]	Sim[13]
Endpoint da VPC da API EKS	Sim	Não	Não
Endpoint público da API do EKS Auth (EKS Pod Identity)	Sim[1]	Sim[1]	Sim[1]
Endpoint da VPC da API do EKS Auth (EKS Pod Identity)	Sim[1]	Sim[1]	Sim[1]
Endpoint público do cluster do Kubernetes	Sim	Não	Não
Endpoint privado do cluster do Kubernetes	Sim[2]	Sim[2]	Não
Sub-redes do cluster do Kubernetes	Sim[2]	Não	Sim[2]
Endereços IP primários do nó	Sim[2]	Não	Sim[2]
Intervalo de CIDR do cluster para endereços IP do Service	Sim[2]	Sim[2]	Não
Endereços IP do Pod do plug-in CNI da VPC	Sim[2]	Sim[2]	Não

Componente	Somente endereços **IPv4**	Somente endereços **IPv6**	Endereços de pilha dupla
URLs do emissor OIDC IRSA	Sim[13]	Sim[13]	Sim[13]

Note

[1] O endpoint é uma pilha dupla com endereços `IPv4` e `IPv6`. As aplicações externas à AWS, os nós do cluster e os pods dentro do cluster podem obter acesso a esse endpoint usando `IPv4` ou `IPv6`.
[2] Você escolhe entre um cluster `IPv4` e um cluster `IPv6` na configuração IP family (`ipFamily`) do cluster ao criar um cluster, e isso não pode ser alterado. Em vez disso, você deve escolher uma configuração diferente ao criar outro cluster e migrar as workloads.
[3] O endpoint de pilha dupla foi introduzido em agosto de 2024. Para usar os endpoints de pilha dupla com a AWS CLI, consulte a configuração dos endpoints de pilha dupla e FIPS no Guia de referência de SDKs e ferramentas da AWS. A seguinte lista contém os novos endpoints:

Endpoint público da API EKS

```
eks.region.api.aws
```

URLs do emissor OIDC IRSA

```
oidc-eks.region.api.aws
```

Requisitos de sub-redes para nós

É possível implantar nós e recursos do Kubernetes nas mesmas sub-redes que você especifica ao criar o cluster. Porém, isso não é necessário. Isso porque também é possível implantar nós e recursos do Kubernetes em sub-redes que você não especificou quando criou o cluster. Se você implantar nós em sub-redes diferentes, o Amazon EKS não criará interfaces de rede de cluster nelas. Qualquer sub-rede na qual você implante nós e recursos do Kubernetes deve atender aos seguintes requisitos:

- As sub-redes devem ter endereços IP disponíveis suficientes para implantar todos os nós e recursos do Kubernetes.
- Se quiser que o Kubernetes atribua endereços IPv6 a Pods e serviços, você deverá ter um bloco CIDR IPv6 e um bloco CIDR IPv4 associados à sub-rede. Para obter mais informações, consulte Associar um bloco CIDR IPv6 à sua sub-rede, no Guia do usuário da Amazon VPC. As tabelas de rotas associadas às sub-redes devem incluir rotas para endereços IPv4 e IPv6. Para obter mais informações, consulte Rotas, no Guia do usuário da Amazon VPC. Pods recebem apenas um endereço IPv6. Porém, as interfaces de rede que criadas pelo Amazon EKS para o seu cluster e os seus nós recebem um endereço IPv4 e um endereço IPv6.
- Se precisar de acesso de entrada pela Internet aos Pods, certifique-se de ter pelo menos uma sub-rede pública com endereços IP disponíveis suficientes para implantar balanceadores de carga e ingressos. Você pode implantar balanceadores de carga em sub-redes públicas. Balanceadores de carga podem balancear carga para Pods em sub-redes privadas ou públicas. Convém implantar nós em sub-redes privadas, se possível.
- Se você planeja implantar nós em uma sub-rede pública, esta deve atribuir automaticamente endereços públicos IPv4 ou endereços IPv6. Se você implantar nós em uma sub-rede privada que tenha um bloco CIDR IPv6 associado, a sub-rede privada também deverá atribuir endereços IPv6 automaticamente. Se você utilizou um modelo de AWS CloudFormation do Amazon EKS para implantar a VPC após 26 de março de 2020, essa configuração estará habilitada. Se você utilizou os modelos para implantar sua VPC antes dessa data ou se utilizar sua própria VPC, deverá habilitar essa configuração manualmente. Para saber mais, consulte Modificar o atributo de endereçamento IPv4 público para sua sub-rede e Modificar o atributo de endereçamento IPv6 para sua sub-rede, no Guia do usuário da Amazon VPC.
- Se a sub-rede na qual você implanta um nó for privada e sua tabela de rotas não incluir uma rota para um dispositivo de conversão de endereços de rede (NAT) (IPv4) ou um gateway somente de saída (IPv6), adicione endpoints de VPC usando o AWS PrivateLink à sua VPC. Endpoints de VPC são necessários para todos os Serviços da AWS com os quais seus nós e Pods precisam se comunicar. Exemplos incluem Amazon ECR, Elastic Load Balancing, Amazon CloudWatch, AWS Security Token Service e Amazon Simple Storage Service (Amazon S3). O endpoint deve incluir a sub-rede na qual os nós se encontram. Nem todos os Serviços da AWS oferecem suporte a endpoints de VPC. Para mais informações, consulte O que é o AWS PrivateLink? e Serviços da AWS que se integram ao AWS PrivateLink. Para obter uma lista de mais requisitos para o Amazon EKS, consulte Implementar clusters privados com acesso limitado à internet.
- Se quiser implantar balanceadores de carga em uma sub-rede, esta deve ter a seguinte etiqueta:
  - Sub-redes privadas

Chave	Valor
kubernetes.io/role/internal-elb	1

- Sub-redes públicas

Chave	Valor
kubernetes.io/role/elb	1

Quando um cluster do Kubernetes da versão 1.18 e anteriores foi criado, o Amazon EKS adicionou a tag a seguir a todas as sub-redes que foram especificadas.

Chave	Valor
kubernetes.io/cluster/ *my-cluster*	shared

Quando você cria um cluster do Kubernetes agora, o Amazon EKS não adiciona a tag às sub-redes. Se a tag estava em sub-redes usadas por um cluster que era previamente uma versão anterior à 1.19, a tag não foi removida automaticamente das sub-redes quando o cluster foi atualizado para uma versão mais recente. Versões 2.1.1 ou anteriores do AWS Load Balancer Controller exigem essa tag. Se você estiver usando uma versão mais recente do controlador do balanceador de carga, poderá remover a tag sem interromper seus serviços.

Se você tiver implantado uma VPC usando eksctl ou qualquer um dos modelos de VPC AWS CloudFormation do Amazon EKS, o seguinte será aplicável:

- Em ou após 26 de março de 2020: os endereços IPv4 públicos serão atribuídos automaticamente por sub-redes públicas aos novos nós implantados em sub-redes públicas.
- Em ou antes de 26 de março de 2020: os endereços IPv4 públicos não serão atribuídos automaticamente por sub-redes públicas aos novos nós implantados em sub-redes públicas.

Essa alteração afeta os novos grupos de nós implantados em sub-redes públicas das seguintes maneiras:

- Grupos de nós gerenciados: se o grupo de nós for implantado em uma sub-rede pública em ou após 22 de abril de 2020, a atribuição automática de endereços IP públicos deverá ser habilitada para a sub-rede pública. Para obter mais informações, consulte Modificar o atributo de endereçamento IPv4 público para a sua sub-rede.
- Grupos de nós autogerenciados Linux, Windows, or Arm: se o grupo de nós for implantado em uma sub-rede pública a partir de 26 de março de 2020, a atribuição automática de endereços IP públicos deverá estar habilitada para a sub-rede pública. Caso contrário, os nós deverão ser iniciados com um endereço IP público. Para obter mais informações, consulte Modificar o atributo de endereçamento IPv4 público para a sua sub-rede ou Atribuir um endereço IPv4 público durante a inicialização da instância.

Requisitos e considerações de sub-redes

Você pode usar Compartilhamento de VPC para compartilhar sub-redes com outras contas da AWS dentro da mesma AWS Organizations. Você pode criar clusters do Amazon EKS em sub-redes compartilhadas com as seguintes considerações:

- O proprietário da sub-rede VPC deve compartilhar uma sub-rede com uma conta de participante antes que essa conta possa criar nela um cluster Amazon EKS.
- Você não pode iniciar recursos usando o grupo de segurança padrão da VPC, pois ele pertence ao proprietário. Além disso, os participantes não podem iniciar recursos usando grupos de segurança de propriedade de outros participantes ou do proprietário.
- Em uma sub-rede compartilhada, o participante e o proprietário controlam separadamente os grupos de segurança na respectiva conta. O proprietário da sub-rede pode visualizar esses grupos de segurança criados pelos participantes, mas não pode realizar neles nenhuma ação. Se o proprietário da sub-rede quiser remover ou modificar esses grupos de segurança, o participante que criou o grupo de segurança deve realizar a ação.
- Se um cluster for criado por um participante, as seguintes considerações se aplicam:
  - A perfil do IAM do cluster e perfis do IAM do nó devem ser criados na conta. Para ter mais informações, consulte Função do IAM do cluster do Amazon EKS e Perfil do IAM em nós do Amazon EKS.
  - Todos os nós devem ser criados pelo mesmo participante, incluindo grupos de nós gerenciados.
- O proprietário da VPC compartilhada não pode visualizar, atualizar ou excluir um cluster criado por um participante na sub-rede compartilhada. Isso é além dos recursos da VPC aos quais

cada conta tem acesso diferente. Para obter mais informações, consulte Responsabilidades e permissões para proprietários e participantes no Manual do usuário do Amazon VPC.

- Se usar o atributo de rede personalizada do Amazon VPC CNI plugin for Kubernetes, você precisará usar os mapeamentos de ID da zona de disponibilidade listados na conta do proprietário para criar cada `ENIConfig`. Para obter mais informações, consulte Rede personalizada para pods.

Para obter informações sobre o compartilhamento de sub-rede da VPC, consulte Compartilhar sua VPC com outras contas no Manual do usuário do Amazon VPC.

Criar uma VPC para o cluster do Amazon EKS

Você pode usar a Amazon Virtual Private Cloud (Amazon VPC) para iniciar recursos da AWS em uma rede virtual definida por você. Essa rede virtual é muito semelhante a uma rede tradicional que pode ser operada no seu próprio datacenter. Porém, ela vem com os benefícios do uso da infraestrutura escalável da Amazon Web Services. É recomendável ter uma profunda compreensão do serviço Amazon VPC antes de implantar clusters do Amazon EKS em ambientes de produção. Para obter mais informações, consulte o Manual do usuário da Amazon VPC.

Um cluster, os nós e os recursos do Kubernetes do Amazon EKS são implantados em uma VPC. Se quiser utilizar uma VPC existente com o Amazon EKS, essa VPC deverá atender aos requisitos descritos em Requisitos e considerações sobre a VPC e a sub-rede do Amazon EKS. Este tópico descreve como criar uma VPC que atende aos requisitos do Amazon EKS utilizando um modelo AWS CloudFormation fornecido pelo Amazon EKS. Depois de implantar um modelo, você pode visualizar os recursos criados por ele para saber exatamente quais recursos foram criados e a configuração desses recursos.

Pré-requisito

Para criar uma VPC para o Amazon EKS, é necessário ter as permissões do IAM necessárias para criar recursos da Amazon VPC. Esses recursos são VPCs, sub-redes, grupos de segurança, tabelas e rotas de rotas e gateways da Internet e de NAT. Para obter mais informações, consulte Criar uma VPC com uma política de exemplo de sub-rede pública, no Guia do usuário da Amazon VPC, e a lista completa de Ações, recursos e chaves de condição do Amazon EC2, na Referência de autorização do serviço.

É possível criar uma VPC com sub-redes públicas e privadas, somente sub-redes públicas ou somente sub-redes privadas.

Public and private subnets

Essa VPC tem duas sub-redes públicas e duas privadas. A tabela de rotas associada a uma sub-rede pública tem uma rota para um gateway da Internet. Porém, a tabela de rotas de uma sub-rede privada não tem uma rota para um gateway da Internet. Uma sub-rede pública e uma privada são implantadas na mesma zona de disponibilidade. As outras sub-redes públicas e privadas são implantadas em uma segunda zona de disponibilidade na mesma Região da AWS. Recomendamos essa opção para a maioria das implantações.

Com essa opção, é possível implantar nós em sub-redes privadas. Essa opção permite que o Kubernetes implante balanceadores de carga nas sub-redes públicas que podem balancear a carga de tráfego para os Pods que são executados em nós nas sub-redes privadas. Endereços `IPv4` públicos são atribuídos automaticamente a nós implantados em sub-redes públicas, mas endereços `IPv4` públicos não são atribuídos a nós implantados a sub-redes privadas.

Você também pode atribuir endereços `IPv6` a nós em sub-redes públicas e privadas. Os nós em sub-redes privadas podem se comunicar com o cluster e outros Serviços da AWS. Pods podem se comunicar com a Internet por meio de um gateway de NAT usando endereços `IPv4` ou de um gateway da Internet somente de saída usando endereços `IPv6` implantados em cada zona de disponibilidade. É implantado um grupo de segurança com regras que negam todo o tráfego de entrada de origens diferentes do cluster ou dos nós, mas permite todo o tráfego de saída. As sub-redes são marcadas para que o Kubernetes possa implantar nelas balanceadores de carga.

Para criar a VPC

1. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.
2. Na barra de navegação, selecione uma Região da AWS que seja compatível com o Amazon EKS.
3. Escolha Create stack (Criar pilha), With new resources (Com novos recursos (padrão)).
4. Em Prerequisite - Prepare template, (Pré-requisito: preparar o modelo), certifique-se de que a opção Template is ready (O modelo está pronto) esteja selecionada. Em seguida, em Specify template (Especificar modelo), selecione Amazon S3 URL. (URL do Simple Storage Service [Amazon S3]).
5. É possível criar uma VPC que ofereça suporte somente a `IPv4` ou uma VPC que ofereça suporte a `IPv4` e a `IPv6`. Cole um dos seguintes URLs na área de texto em Amazon S3 URL (URL do Amazon S3) e escolha Next (Próximo):

- IPv4

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
amazon-eks-vpc-private-subnets.yaml
```

- IPv4 e IPv6

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
amazon-eks-ipv6-vpc-public-private-subnets.yaml
```

6. Na página Specify stack details (Especificar detalhes da pilha), insira os parâmetros e escolha Next (Próximo).

   - Stack name (Nome da pilha): escolha um nome de pilha para a pilha do AWS CloudFormation. Por exemplo, você pode usar o nome do modelo empregado na etapa anterior. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
   - VpcBlock: escolha um intervalo de CIDR IPv4 para a sua VPC. Cada nó, Pod e balanceador de carga implantado recebe um endereço IPv4 desse bloco. Os valores padrão de IPv4 fornecem endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-los. Para obter mais informações, consulte VPC and subnet sizing (Dimensionamento da VPC e da sub-rede) no Guia do usuário da Amazon VPC. Você também pode adicionar blocos CIDR adicionais à VPC depois que ela for criada. Se estiver criando uma VPC IPv6, os intervalos de CIDR IPv6 serão automaticamente atribuídos a você no espaço do endereço unicast global da Amazon.
   - PublicSubnet01Block: especifique um bloco CIDR IPv4 para a sub-rede pública 1. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo. Se você estiver criando uma VPC IPv6, esse bloco será especificado para você no modelo.
   - PublicSubnet02Block: especifique um bloco CIDR IPv4 para a sub-rede pública 2. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo. Se você estiver criando uma VPC IPv6, esse bloco será especificado para você no modelo.
   - PrivateSubnet01Block: especifique um bloco CIDR IPv4 para a sub-rede privada 1. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se

isso não acontecer, você poderá alterá-lo. Se você estiver criando uma VPC `IPv6`, esse bloco será especificado para você no modelo.

- PrivateSubnet02Block: especifique um bloco CIDR `IPv4` para a sub-rede privada 2. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo. Se você estiver criando uma VPC `IPv6`, esse bloco será especificado para você no modelo.

7. (Opcional) Na página Configure stack options (Configurar opções de pilha), etiquete os seus recursos de pilha e, em seguida, escolha Next (Próximo).
8. Na página Review (Revisão), escolha Create stack (Criar pilha).
9. Quando a pilha estiver criada, selecione-a no console e escolha Outputs (Saídas).
10. Registre o VpcId para a VPC que foi criada. Você precisará disso ao criar seu cluster e nós.
11. Registre os SubnetIds para as sub-redes que foram criadas e se você as criou como sub-redes públicas ou privadas. É necessário pelo menos dois deles ao criar seu cluster e os nós.
12. Se você criou uma VPC `IPv4`, ignore essa etapa. Se você criou uma VPC `IPv6`, será necessário habilitar a opção de atribuição automática de endereço `IPv6` para as sub-redes públicas criadas pelo modelo. Essa configuração já está habilitada para as sub-redes privadas. Conclua as etapas a seguir para habilitar as configurações:

    a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.

    b. No painel de navegação à esquerda, escolha Subnets (Sub-redes).

    c. Selecione uma de suas sub-redes públicas (***stack-name***/SubnetPublic01 ou ***stack-name***/SubnetPublic02, que contêm a palavra public) e escolha Actions (Ações) e, em seguida, Edit subnet settings (Editar configurações de sub-redes).

    d. Escolha a opção Habilitar atribuição automática do endereço **IPv6** e clique em Salvar.

    e. Conclua as etapas anteriores novamente para a sua outra sub-rede pública.

Only public subnets

Esta VPC tem três sub-redes públicas que são implantadas em zonas de disponibilidade diferentes em uma Região da AWS. Todos os nós recebem endereços `IPv4` públicos automaticamente e podem enviar e receber tráfego de Internet por meio de um gateway da Internet. É implantado um grupo de segurança que nega todo o tráfego de entrada e permite todo o tráfego de saída. As sub-redes são marcadas para que o Kubernetes possa implantar nelas balanceadores de carga.

Para criar a VPC

1. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.
2. Na barra de navegação, selecione uma Região da AWS que seja compatível com o Amazon EKS.
3. Escolha Create stack (Criar pilha), With new resources (Com novos recursos (padrão)).
4. Em Prepare template (Preparar o template), verifique se a opção Template is ready (O template está pronto) está selecionada e, em Template source (Origem do template), selecione Amazon S3 URL (URL do Amazon S3).
5. Cole o seguinte URL na área de texto em Amazon S3 URL (URL do Amazon S3) e escolha Next (Próximo):

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

6. Na página Specify Details (Especificar detalhes), insira os parâmetros e escolha Next (Próximo).

   - Stack name (Nome da pilha): escolha um nome de pilha para a pilha do AWS CloudFormation. Por exemplo, você pode chamar de ***amazon-eks-vpc-sample***. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
   - VpcBlock: escolha um bloco CIDR para a VPC. Cada nó, Pod e balanceador de carga implantado recebe um endereço `IPv4` desse bloco. Os valores padrão de `IPv4` fornecem endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-los. Para obter mais informações, consulte VPC and subnet sizing (Dimensionamento da VPC e da sub-rede) no Guia do usuário da Amazon VPC. Você também pode adicionar blocos CIDR adicionais à VPC depois que ela for criada.
   - Subnet01Block: especifique um bloco CIDR para a sub-rede 1. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.

- Subnet02Block: especifique um bloco CIDR para a sub-rede 2. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.
- Subnet03Block: especifique um bloco CIDR para a sub-rede 3. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.

7. (Opcional) Na página Options (Opções), marque os recursos da pilha. Escolha Próximo.
8. Na página Review (Revisar), escolha Create (Criar).
9. Quando a pilha estiver criada, selecione-a no console e escolha Outputs (Saídas).
10. Registre o VpcId para a VPC que foi criada. Você precisará disso ao criar seu cluster e nós.
11. Registre os SubnetIds para as sub-redes que foram criadas. É necessário pelo menos dois deles ao criar seu cluster e os nós.
12. (Opcional) Todos os clusters que você implantar nessa VPC podem atribuir endereços `IPv4` privados aos seus Pods e services. Se quiser implantar clusters nessa VPC para atribuir endereços `IPv6` privados aos seus Pods e services, faça atualizações na VPC, na sub-rede, em tabelas de rotas e em grupos de segurança. Para obter mais informações, consulte Migrar VPCs existentes de `IPv4` para `IPv6`, no Guia do usuário da Amazon VPC. O Amazon EKS exige que as suas sub-redes stejam com a opção de `Auto-assign` de endereços `IPv6` habilitada. Por padrão, ele está desabilitado.

Only private subnets

Esta VPC tem três sub-redes privadas que são implantadas em zonas de disponibilidade diferentes na Região da AWS. Os recursos implantados nas sub-redes não podem acessar a Internet, nem a internet pode acessar os recursos nas sub-redes. O modelo cria endpoints de VPC utilizando o AWS PrivateLink para vários Serviços da AWS que os nós normalmente precisam acessar. Se seus nós precisarem de acesso à Internet de saída, você poderá adicionar um gateway de NAT público na Zona de disponibilidade de cada sub-rede após a criação da VPC. É criado um grupo de segurança que nega todo o tráfego de entrada, exceto de recursos implantados nas sub-redes. Um grupo de segurança também permite todo o tráfego de saída. As sub-redes são marcadas para que o Kubernetes possa implantar nelas balanceadores de carga internos. Se estiver criando uma VPC com essa configuração, consulte Implementar clusters privados com acesso limitado à internet para conhecer requisitos e considerações adicionais.

Para criar a VPC

1. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.
2. Na barra de navegação, selecione uma Região da AWS que seja compatível com o Amazon EKS.
3. Escolha Create stack (Criar pilha), With new resources (Com novos recursos (padrão)).
4. Em Prepare template (Preparar o template), verifique se a opção Template is ready (O template está pronto) está selecionada e, em Template source (Origem do template), selecione Amazon S3 URL (URL do Amazon S3).
5. Cole o seguinte URL na área de texto em Amazon S3 URL (URL do Amazon S3) e escolha Next (Próximo):

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-fully-private-vpc.yaml
```

6. Na página Specify Details (Especificar detalhes), insira os parâmetros e escolha Next (Próximo).

   - Stack name (Nome da pilha): escolha um nome de pilha para a pilha do AWS CloudFormation. Por exemplo, você pode chamar de ***amazon-eks-fully-private-vpc***. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
   - VpcBlock: escolha um bloco CIDR para a VPC. Cada nó, Pod e balanceador de carga implantado recebe um endereço `IPv4` desse bloco. Os valores padrão de `IPv4` fornecem endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-los. Para obter mais informações, consulte VPC and subnet sizing (Dimensionamento da VPC e da sub-rede) no Guia do usuário da Amazon VPC. Você também pode adicionar blocos CIDR adicionais à VPC depois que ela for criada.
   - PrivateSubnet01Block: especifique um bloco CIDR para a sub-rede 1. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.

- PrivateSubnet02Block: especifique um bloco CIDR para a sub-rede 2. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.
- PrivateSubnet03Block: especifique um bloco CIDR para a sub-rede 3. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.

7. (Opcional) Na página Options (Opções), marque os recursos da pilha. Escolha Próximo.
8. Na página Review (Revisar), escolha Create (Criar).
9. Quando a pilha estiver criada, selecione-a no console e escolha Outputs (Saídas).
10. Registre o VpcId para a VPC que foi criada. Você precisará disso ao criar seu cluster e nós.
11. Registre os SubnetIds para as sub-redes que foram criadas. É necessário pelo menos dois deles ao criar seu cluster e os nós.
12. (Opcional) Todos os clusters que você implantar nessa VPC podem atribuir endereços `IPv4` privados aos seus Pods e services. Se quiser implantar clusters nessa VPC para atribuir endereços `IPv6` privados aos seus Pods e services, faça atualizações na VPC, na sub-rede, em tabelas de rotas e em grupos de segurança. Para obter mais informações, consulte Migrar VPCs existentes de `IPv4` para `IPv6`, no Guia do usuário da Amazon VPC. O Amazon EKS exige que as suas sub-redes stejam com a opção de `Auto-assign` de endereços `IPv6` habilitada (é desabilitada por padrão).

Considerações e requisitos sobre grupos de segurança do Amazon EKS

Este tópico descreve os requisitos de grupos de segurança de um cluster do Amazon EKS.

Ao criar um cluster, o Amazon EKS cria um grupo de segurança com o nome `eks-cluster-sg-`*`my-cluster`*`-`*`uniqueID`*. Esse grupo de segurança tem as seguintes regras padrão:

Tipo de regra	Protocolo	Portas	Origem	Destino
Entrada	Todos	Todos	Self	
Saída	Todos	Todos		0.0.0.0/0 (`IPv4`) ou ::/0 (`IPv6`)

> **Important**
>
> Se o cluster não precisar da regra de saída, você poderá removê-la. Se você a remover, ainda deverá ter as regras mínimas listadas em Restringir o tráfego do cluster. Se você remover a regra de entrada, o Amazon EKS a recriará sempre que o cluster for atualizado.

O Amazon EKS adiciona as tags a seguir ao grupo de segurança. Se você remover as tags, o Amazon EKS as adicionará novamente ao grupo de segurança sempre que o cluster for atualizado.

Chave	Valor
`kubernetes.io/cluster/ my-cluster`	`owned`
`aws:eks:cluster-name`	*my-cluster*
`Name`	`eks-cluster-sg-` *my-cluste r* `-uniqueid`

O Amazon EKS associa automaticamente esse grupo de segurança aos recursos a seguir, que ele também cria:

- Duas a quatro interfaces de rede elásticas (chamadas no restante deste documento de interfaces de rede) que são criadas no momento em que você cria seu cluster.
- Interfaces de rede dos nós em qualquer grupo de nós gerenciados que você cria.

As regras padrão permitem que todo o tráfego flua livremente entre o cluster e os nós e aceitam todo o tráfego de saída para qualquer destino. Ao criar um cluster, você tem a opção de especificar seus próprios grupos de segurança. Se fizer isso, o Amazon EKS também associará os grupos de segurança especificados às interfaces de rede que ele criar para o cluster. Porém, ele não os associa a nenhum grupo de nós que você criar.

É possível determinar o ID do grupo de segurança do cluster no AWS Management Console, na seção Networking (Redes) do cluster. Ou, você pode fazer isso executando o seguinte comando AWS CLI:

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Restringir o tráfego do cluster

Se for necessário limitar as portas abertas entre o cluster e os nós, você poderá remover as regras de saída padrão e adicionar as seguintes regras mínimas que são necessárias para o cluster. Se você remover a regra de entrada padrão, o Amazon EKS a recriará sempre que o cluster for atualizado.

Tipo de regra	Protocolo	Port	Destino
Saída	TCP	443	Security group de cluster
Saída	TCP	10250	Security group de cluster
Saída (DNS)	TCP e UDP	53	Security group de cluster

Também é necessário adicionar regras para o seguinte tráfego:

- Qualquer protocolo e as portas que você espera que os nós usem para comunicação entre eles.
- Acesso de saída à Internet, para que os nós possam acessar as APIs do Amazon EKS para introspecção de clusters e registro de nós no momento da execução. Se os nós não tiverem acesso à internet, confira Implementar clusters privados com acesso limitado à internet para conhecer considerações adicionais.
- Acesso ao nó para obter imagens de contêiner do Amazon ECR ou de outras APIs de registros de contêiner dos quais eles precisem extrair imagens, p. ex., DockerHub. Para obter mais informações, consulte Intervalos de endereços IP da AWS no Referência geral da AWS.
- Acesso do nó ao Amazon S3
- Regras distintas são necessárias para endereços `IPv4` e `IPv6`.

Se estiver pensando em limitar regras, recomendamos testar completamente todos os seus Pods antes de aplicar as regras modificadas a um cluster de produção.

Se você tiver implantado originalmente um cluster com o Kubernetes `1.14` e uma versão `eks.3` ou anterior da plataforma, considere os pontos a seguir.

- Também é possível ter grupos de segurança e nós do ambiente de gerenciamento. Quando esses grupos foram criados, eles incluíam as regras restritas listadas na tabela anterior. Esses grupos de segurança não são mais necessários e podem ser removidos. Porém, você precisa garantir que seu grupo de segurança de cluster contenha as regras que esses grupos contêm.
- Se você implantou o cluster utilizando a API diretamente ou utilizou uma ferramenta como AWS CLI ou AWS CloudFormation para criá-lo e não especificou um grupo de segurança na criação do cluster, o grupo de segurança padrão da VPC foi aplicado às interfaces de rede do cluster que foram criadas pelo Amazon EKS.

Complementos de redes do Amazon EKS

O Amazon EEKS está disponível para seu cluster do Amazon EKS.

Complementos integrados

> **Note**
>
> Se você criar clusters de qualquer forma, exceto usando o console, cada cluster virá com as versões autogerenciadas dos complementos integrados. As versões autogerenciadas não podem ser gerenciadas usando o AWS Management Console, a AWS Command Line Interface ou SDKs. Você gerencia a configuração e os upgrades dos complementos autogerenciados.
> Recomendamos adicionar o tipo Amazon EKS do complemento ao seu cluster em vez de usar o tipo autogerenciado do complemento. Se criar clusters no console, esses complementos serão instalados com o tipo do Amazon EKS.

Amazon VPC CNI plugin for Kubernetes

Esse complemento da CNI cria interfaces de rede elásticas e as anexa aos nós do Amazon EC2. O complemento também atribui um endereço `IPv4` ou `IPv6` privado da sua VPC a cada Pod e serviço. Esse complemento é instalado no cluster por padrão. Para ter mais informações, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

CoreDNS

O CoreDNS é um servidor DNS flexível e extensível que pode atuar como DNS de cluster do Kubernetes. O CoreDNS fornece resolução de nomes para todos os Pods no cluster. Esse complemento é instalado no cluster por padrão. Para ter mais informações, consulte Trabalhando com o complemento CoreDNS do Amazon EKS.

`kube-proxy`

Esse complemento mantém as regras de rede em seus nós do Amazon EC2 e viabiliza a comunicação de rede com seus Pods. Esse complemento é instalado no cluster por padrão. Para ter mais informações, consulte Trabalhando com o complemento `kube-proxy` do Kubernetes.

Complementos opcionais de rede da AWS

AWS Load Balancer Controller

Ao implantar objetos de serviço do Kubernetes do tipo `loadbalancer`, o controlador cria Network Load Balancers da AWS. Quando você cria objetos de entrada do Kubernetes, o controlador cria Application Load Balancers da AWS. Recomendamos usar esse controlador para provisionar Network Load Balancers em vez de usar o controlador legado do Cloud Provider incorporado no Kubernetes. Para obter mais informações, consulte a documentação do AWS Load Balancer Controller.

Controlador de API de gateway da AWS

Esse controlador permite conectar serviços entre vários clusters do Kubernetes usando a API de gateway do Kubernetes. O controlador conecta serviços do Kubernetes executados em instâncias, contêineres e funções sem servidor do Amazon EC2 usando o serviço Amazon VPC Lattice. Para obter mais informações, consulte a documentação do AWS Gateway API Controller

Para obter mais informações sobre complementos, consulte Complementos do Amazon EKS.

Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS

O complemento Amazon VPC CNI plugin for Kubernetes é implantado em cada nó do Amazon EC2 em seu cluster do Amazon EKS. O complemento cria interfaces de rede elásticas e as anexa aos nós do Amazon EC2. O complemento também atribui um endereço `IPv4` ou `IPv6` privado da sua VPC a cada Pod e serviço.

Uma versão do complemento é implantada com cada nó do Fargate em seu cluster, mas você não a atualiza em nós do Fargate. Há outros plug-ins de CNI compatíveis disponíveis para uso em clusters do Amazon EKS, mas esse é o único plug-in de CNI compatível com o Amazon EKS.

A tabela a seguir lista a versão mais recente do complemento do Amazon EKS disponível para cada versão do Kubernetes.

Versão do Kubernetes	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
Tipo Amazon EKS de versão da CNI da VPC	v1.18.3-eksbuil	v1.18.3-eksbuil	v1.18.3-eksbuil	v1.18.3-eksbuil	v1.18.3-eksbuil	v1.18.3-eksbuil	v1.18.3-eksbuil	v1.18.3-eksbuild.1

> **Important**
>
> Se você estiver gerenciando esse complemento automaticamente, as versões na tabela podem não ser as mesmas que as versões autogerenciadas disponíveis. Para obter mais informações sobre como atualizar o tipo autogerenciado desse complemento, consulte Atualização do complemento autogerenciado.

> **Important**
>
> Para atualizar para a CNI da VPC v1.12.0 ou posterior, primeiro é necessário fazer o upgrade para a CNI da VPC v1.7.0. Convém atualizar uma versão secundária de cada vez.

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS.
- Um provedor OpenID Connect (OIDC) do AWS Identity and Access Management (IAM) existente para o cluster. Para determinar se você tem ou para criar uma, consulte Criar um provedor OIDC do IAM para o cluster.

- Um perfil do IAM com a política do IAM AmazonEKS_CNI_Policy (se o cluster usar a família `IPv4`) ou uma política IPv6 (se o seu cluster usar a família `IPv6`) anexada a ela. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).
- Se você estiver usando a versão `1.7.0` ou posterior do Amazon VPC CNI plugin for Kubernetes e usar políticas de segurança de Pod personalizadas, consulte Excluir política de segurança de Pod padrão do Amazon EKSPolítica de segurança de pods.
-

  Important

  As versões do Kubernetes `v1.16.0` do Amazon VPC CNI plugin for para `v1.16.1` removeram a compatibilidade com as versões `1.23` e anteriores do Kubernetes. A versão VPC CNI `v1.16.2` restaura a compatibilidade com as versões `1.23` e anteriores do Kubernetes com a especificação CNI `v0.4.0`.

  As versões `v1.16.0` do Amazon VPC CNI plugin for do Kubernetes para `v1.16.1` implementam a versão `v1.0.0` da especificação CNI. A especificação CNI `v1.0.0` tem suporte de clusters do EKS que executam as versões `v1.24` ou posteriores do Kubernetes. A versão `v1.16.0` a `v1.16.1` da VPC CNI e a especificação CNI `v1.0.0` não são compatíveis com a versão `v1.23` ou posterior do Kubernetes. Para obter mais informações sobre a versão `v1.0.0` da especificação CNI, consulte Container Network Interface (CNI) Specification em

Considerações

- As versões são especificadas como `major-version.minor-version.patch-version-eksbuild.build-number`.
- Verifique a compatibilidade de versão para cada atributo

  Alguns recursos de cada versão do Kubernetes do Amazon VPC CNI plugin for exigem determinadas versões do Kubernetes. Ao utilizar diferentes recursos do Amazon EKS, se uma versão específica do complemento for necessária, então ela será anotada na documentação de recursos. A menos que você tenha um motivo específico para executar uma versão anterior, recomendamos que execute a versão mais recente.

Criar o complemento do Amazon EKS

Crie o tipo do Amazon EKS do complemento.

1. Veja qual versão do complemento está atualmente instalada no cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

   Veja um exemplo de saída abaixo.

```
v1.16.4-eksbuild.2
```

2. Veja qual tipo de complemento está atualmente instalado no cluster. Dependendo da ferramenta com a qual você criou o cluster, talvez você não tenha o tipo de complemento do Amazon EKS instalado em seu cluster atualmente. Substitua *my-cluster* pelo nome do cluster.

```
$ aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

   Se um número de versão for retornado, você tem o tipo de complemento do Amazon EKS instalado no cluster, e não precisa completar as etapas restantes deste procedimento. Se um erro for retornado, você não tem o tipo de complemento do Amazon EKS instalado no cluster. Conclua as etapas restantes desse procedimento para instalá-lo.

3. Salve a configuração do complemento instalado atualmente.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. Crie o complemento usando o AWS CLI. Se você quiser usar o AWS Management Console ou `eksctl` para criar o complemento, consulte Criar um complemento do Amazon EKS e especifique `vpc-cni` como o nome do complemento. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.

   - Substitua o *my-cluster* pelo nome do cluster.
   - Substitua *v1.18.3-eksbuild.1* pela versão mais recente listada na tabela de versões mais recentes da versão do cluster.

- Substitua *111122223333* pelo ID da sua conta e *AmazonEKSVPCCNIRole* pelo nome do perfil do IAM existente que você criou. A especificação de um perfil exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte Criar um provedor OIDC do IAM para o cluster.

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.18.3-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
```

Se você aplicou configurações personalizadas ao seu complemento atual que entrem em conflito com as configurações padrão do complemento Amazon EKS, a criação poderá falhar. Se a criação falhar, você receberá um erro que poderá ajudar a resolver o problema. Como alternativa, você pode adicionar **`--resolve-conflicts OVERWRITE`** ao comando anterior. Isso permite que o complemento substitua todas as configurações personalizadas existentes. Depois de criar o complemento, você pode atualizá-lo com suas configurações personalizadas.

5. Confirme se a versão mais recente do complemento para a versão Kubernetes do seu cluster foi adicionada ao cluster. Substitua o *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

Pode levar alguns segundos para que a criação do complemento seja concluída.

Veja um exemplo de saída abaixo.

```
v1.18.3-eksbuild.1
```

6. Se você fez configurações personalizadas no complemento original, antes de criar o complemento Amazon EKS, use a configuração que você salvou em uma etapa anterior para atualizar o complemento Amazon EKS com as configurações personalizadas.
7. (Opcional) Instale `cni-metrics-helper` em seu cluster. Ele extrai a interface de rede elástica e as informações de endereço IP, agrega-as em um nível de cluster e publica as métricas no Amazon CloudWatch. Para obter mais informações, consulte cni-metrics-helper no GitHub.

Atualizar o complemento do Amazon EKS

Atualize o tipo do Amazon EKS do complemento. Se você não adicionou o tipo Amazon EKS do complemento ao cluster, adicione-o ou consulte Atualização do complemento autogerenciado, em vez de concluir esse procedimento.

1. Veja qual versão do complemento está atualmente instalada no cluster. Substitua *my-cluster* pelo nome do cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    "addon.addonVersion" --output text
   ```

   Veja um exemplo de saída abaixo.

   ```
   v1.16.4-eksbuild.2
   ```

   Se a versão retornada for a mesma da versão do Kubernetes do cluster na tabela de versões mais recente, você já tem a versão mais recente instalada no cluster e não precisa concluir o restante desse procedimento. Se você receber um erro, em vez de um número de versão no resultado, você não tem o tipo Amazon EKS do complemento instalado no cluster. Você precisa criar o complemento antes de poder atualizá-lo com este procedimento.
2. Salve a configuração do complemento instalado atualmente.

   ```
   kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
   ```
3. Atualize o complemento usando a AWS CLI. Se você quiser usar o AWS Management Console ou `eksctl` para atualizar o complemento, consulte Atualizar um complemento do Amazon EKS. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.

   - Substitua o *my-cluster* pelo nome do cluster.
   - Substitua *v1.18.3-eksbuild.1* pela versão mais recente listada na tabela de versões mais recentes da versão do cluster.
   - Substitua *111122223333* pelo ID da sua conta e *AmazonEKSVPCCNIRole* pelo nome do perfil do IAM existente que você criou. A especificação de um perfil exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte Criar um provedor OIDC do IAM para o cluster.

- A opção *PRESERVE* de **--resolve-conflicts** mantém os valores de configuração existentes para o complemento. Se você definiu valores personalizados para as configurações do complemento e não usar essa opção, o Amazon EKS sobrescreverá seus valores pelos valores padrão. Se você usar essa opção, recomendamos testar qualquer alteração de campo e valor em um cluster que não seja de produção antes de atualizar o complemento no cluster de produção. Se você alterar esse valor para `OVERWRITE`, todas as configurações serão alteradas para os valores padrão do Amazon EKS. Se você definiu valores personalizados para qualquer configuração, eles poderão ser sobrescritos pelos valores padrão do Amazon EKS. Se você alterar esse valor para `none`, o Amazon EKS não alterará o valor de nenhuma configuração, mas a atualização poderá falhar. Se a atualização falhar, você receberá uma mensagem de erro para ajudar a resolver o conflito.
- Se você não estiver atualizando uma configuração, remova **--configuration-values '{*"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}*}'** do comando. Se você estiver atualizando uma configuração, substitua *"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}* pela configuração que você quiser definir. Neste exemplo, a variável de ambiente `AWS_VPC_K8S_CNI_EXTERNALSNAT` é definida como `true`. O valor que você especificar deve ser válido para o esquema da configuração. Se não souber qual é o esquema de configuração, execute **aws eks describe-addon-configuration --addon-name vpc-cni --addon-version *v1.18.3-eksbuild.1***, substituindo *v1.18.3-eksbuild.1* pelo número da versão do complemento cuja configuração você deseja ver. O esquema é retornado na saída. Se você tiver alguma configuração personalizada existente, quiser remover tudo e definir os valores de todas as configurações de volta aos padrões do Amazon EKS, remova *"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}* do comando, para que **{}** fique vazio. Para obter uma explicação de cada configuração, consulte Variáveis de configuração CNI no GitHub.

  ```
  aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
  version v1.18.3-eksbuild.1 \
      --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
   \
      --resolve-conflicts PRESERVE --configuration-values '{"env":
  {"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'
  ```

  Pode levar alguns segundos para que a atualização seja concluída.

4. Confirme se a versão do complemento foi atualizada. Substitua o *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

Pode levar alguns segundos para que a atualização seja concluída.

Veja um exemplo de saída abaixo.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.18.3-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/vpc-
cni/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
        "createdAt": "2023-04-12T18:25:19.319000+00:00",
        "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
        "serviceAccountRoleArn":
 "arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole",
        "tags": {},
        "configurationValues": "{\"env\":{\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":\"true
\"}}"
    }
}
```

Atualização do complemento autogerenciado

> **Important**
>
> Recomendamos adicionar o tipo Amazon EKS do complemento ao seu cluster em vez de usar o tipo autogerenciado do complemento. Se você não estiver familiarizado com a diferença entre os tipos, consulte the section called “Complementos do Amazon EKS”. Para obter mais informações sobre como adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criar um complemento”. Se você não conseguir usar o complemento do Amazon EKS, recomendamos que você envie um problema sobre o motivo pelo qual não pode usar o repositório GitHub para roteiro de contêineres.

1. Confirme que você não tem o tipo de complemento Amazon EKS instalado em seu cluster. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

   Se uma mensagem erro for retornada, você não tem o tipo de complemento do Amazon EKS instalado no cluster. Para gerenciar automaticamente o complemento, conclua as etapas restantes neste procedimento para atualizar o complemento. Se receber um número de versão, você tem o tipo de complemento do Amazon EKS instalado no cluster. Para atualizá-lo, use o procedimento em Atualizar um complemento do Amazon EKS, em vez de usar este procedimento. Se não estiver familiarizado com a diferença entre os tipos de complemento, consulte Complementos do Amazon EKS.

2. Veja qual versão da imagem do contêiner está atualmente instalada em seu cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

   Veja um exemplo de saída abaixo.

```
v1.16.4-eksbuild.2
```

   O resultado pode não incluir o número da compilação.

3. Faça o backup de suas configurações atuais para que você possa definir as mesmas configurações após atualizar sua versão.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. Para examinar as versões disponíveis e se familiarizar com as alterações feitas na versão para a qual você deseja atualizar, consulte `releases` no GitHub. Observe que recomendamos atualizar para a mesma versão `major.minor.patch` listada na tabela de versões mais recentes disponíveis, mesmo se versões posteriores estiverem disponíveis no GitHub. As versões de compilação listadas na tabela não são especificadas nas versões autogerenciadas listadas no GitHub. Atualize a versão concluindo as tarefas em uma das seguintes opções:

- Se você não tiver nenhuma configuração personalizada para o complemento, execute o comando abaixo do cabeçalho `To apply this release:` no GitHub específico da versão para a qual você está atualizando.
- Se você tiver configurações personalizadas, baixe o manifesto com o comando a seguir. Altere *`https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.18.3/config/master/aws-k8s-cni.yaml`* para a URL da versão no GitHub para a qual você estiver atualizando.

```
curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.18.3/config/master/aws-k8s-cni.yaml
```

Se necessário, modifique o manifesto com as configurações personalizadas do backup que você fez em uma etapa anterior e aplique o manifesto modificado ao cluster. Se seus nós não tiverem acesso aos repositórios privados do Amazon ECR do Amazon EKS dos quais as imagens são extraídas (veja as linhas que começam com `image:` no manifesto), você precisará baixar as imagens, copiá-las para seu próprio repositório e modificar o manifesto a fim de extrair as imagens do seu repositório. Para obter mais informações, consulte Copiar uma imagem de contêiner de um repositório para outro.

```
kubectl apply -f aws-k8s-cni.yaml
```

5. Confirme se agora a nova versão está instalada em seu cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: | cut -d : -f 3
```

Veja um exemplo de saída abaixo.

```
v1.18.3
```

6. (Opcional) Instale `cni-metrics-helper` em seu cluster. Ele extrai a interface de rede elástica e as informações de endereço IP, agrega-as em um nível de cluster e publica as métricas no Amazon CloudWatch. Para obter mais informações, consulte cni-metrics-helper no GitHub.

Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA)

O Amazon VPC CNI plugin for Kubernetes é o plugin de rede para redes de Pod em clusters do Amazon EKS. O plug-in é responsável por alocar os endereços IP da VPC para os nós do Kubernetes e configurar as redes necessárias para os Pods em cada nó. O plugin:

- Ele exige permissões AWS Identity and Access Management (IAM). Se o cluster usa a família IPv4, as permissões são especificadas na política `AmazonEKS_CNI_Policy` gerenciada pela AWS. Se o seu cluster utiliza a família IPv6, as permissões devem ser adicionadas a um política do IAM que você mesmo cria.É possível anexar a política a Amazon EKS node IAM role ou a uma função do IAM separada. Recomendamos atribuir a uma função separada, conforme detalhado neste tópico.
- Cria e está configurado para usar uma conta de serviço Kubernetes chamada `aws-node` quando ele for implantado. A conta de serviço está vinculada a um `clusterrole` do Kubernetes chamado `aws-node`, ao qual são atribuídas as permissões necessárias do Kubernetes.

> **Note**
>
> Os Pods para o Amazon VPC CNI plugin for Kubernetes têm acesso às permissões atribuídas a Amazon EKS node IAM role, a menos que você bloqueie o acesso ao IMDS. Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS.
- Um provedor OpenID Connect (OIDC) do AWS Identity and Access Management (IAM) existente para o cluster. Para determinar se você tem ou para criar uma, consulte Criar um provedor OIDC do IAM para o cluster.

Etapa 1: Criar o perfil do IAM Amazon VPC CNI plugin for Kubernetes

Como criar o perfil do IAM

1. Determine a família de IP do seu cluster.

```
aws eks describe-cluster --name my-cluster | grep ipFamily
```

Veja um exemplo de saída abaixo.

```
"ipFamily": "ipv4"
```

Em vez disso, a saída pode retornar `ipv6`.

2. Crie o perfil do IAM. Você pode usar `eksctl` ou `kubectl` e a AWS CLI para criar seu perfil do IAM.

   eksctl

   Crie um perfil do IAM e anexe a política do IAM a esse perfil por meio do comando que corresponde à família de IP do cluster. O comando cria e implanta uma pilha do AWS CloudFormation que cria um perfil do IAM, anexa a ela a política que você especificar e anota a conta de serviço do `aws-node` Kubernetes existente com o ARN do perfil do IAM criada.

   - `IPv4`

     Substitua *my-cluster* pelos seus próprios valores.

     ```
     eksctl create iamserviceaccount \
         --name aws-node \
         --namespace kube-system \
         --cluster my-cluster \
         --role-name AmazonEKSVPCCNIRole \
         --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
         --override-existing-serviceaccounts \
         --approve
     ```

   - `IPv6`

     Substitua *my-cluster* pelos seus próprios valores. Substitua *111122223333* pelo ID da conta e *AmazonEKS_CNI_IPv6_Policy* pelo nome da política de `IPv6`. Se não tiver uma política `IPv6`, consulte Criar uma política do IAM para clusters que usam a família `IPv6` para criá-la. Para utilizar `IPv6` com seu cluster, este deve atender a vários requisitos. Para ter mais informações, consulte Endereços `IPv6` para clusters, Pods e services.

```
eksctl create iamserviceaccount \
    --name aws-node \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKSVPCCNIRole \
    --attach-policy-arn
 arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
    --override-existing-serviceaccounts \
    --approve
```

kubectl and the AWS CLI

1. Exiba a URL do provedor OIDC do cluster.

   ```
   aws eks describe-cluster --name my-cluster --query
    "cluster.identity.oidc.issuer" --output text
   ```

   Veja um exemplo de saída abaixo.

   ```
   https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
   ```

   Se nenhum resultado for retornado, você deverá criar um provedor OIDC do IAM do seu cluster.

2. Copie o conteúdo a seguir em um arquivo denominado *vpc-cni-trust-policy.json*. Substitua *111122223333* pelo ID da conta e *EXAMPLED539D4633E53DE1B71EXAMPLE* pela saída retornada na etapa anterior. Substitua *region-code* pela Região da AWS em que está o cluster.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Federated": "arn:aws:iam::111122223333:oidc-provider/
   oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
               },
               "Action": "sts:AssumeRoleWithWebIdentity",
   ```

```
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:aws-node"
                }
            }
        }
    ]
}
```

3. Crie a função. É possível substituir *AmazonEKSVPCCNIRole* por qualquer nome que você escolher.

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

4. Anexe a política do IAM necessária à função. Execute o comando que corresponde à família de IP do seu cluster.

   - IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSVPCCNIRole
```

   - IPv6

     Substitua *111122223333* pelo ID da conta e *AmazonEKS_CNI_IPv6_Policy* pelo nome da política de IPv6. Se não tiver uma política IPv6, consulte Criar uma política do IAM para clusters que usam a família IPv6 para criá-la. Para utilizar IPv6 com seu cluster, este deve atender a vários requisitos. Para ter mais informações, consulte Endereços IPv6 para clusters, Pods e services.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSVPCCNIRole
```

5. Execute o seguinte comando para anotar a conta de serviço `aws-node` com o ARN do perfil do IAM criado por você anteriormente: Substitua *example values* pelos seus próprios valores.

```
kubectl annotate serviceaccount \
    -n kube-system aws-node \
    eks.amazonaws.com/role-
arn=arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
```

3. (Opcional) Configure o tipo de endpoint do AWS Security Token Service utilizado pela sua conta de serviço Kubernetes. Para ter mais informações, consulte Configurar o endpoint do AWS Security Token Service para uma conta de serviço.

Etapa 2: Reimplantar os Pods do Amazon VPC CNI plugin for Kubernetes

1. Exclua e recrie os Pods existentes associados à conta de serviço para aplicar as variáveis de ambiente de credenciais. A anotação não é aplicada a Pods que estejam sendo executados sem a anotação. O comando a seguir exclui os `aws-node` DaemonSet Pods existentes e os implanta com a anotação da conta de serviço.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

2. Confirme se todos os Pods foram reiniciados.

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. Descreva um dos Pods e verifique se as variáveis de ambiente `AWS_WEB_IDENTITY_TOKEN_FILE` e `AWS_ROLE_ARN` existem. Substitua *cpjw7* pelo nome de um dos Pods retornados no resultado da etapa anterior.

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\|
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

Veja um exemplo de saída abaixo.

```
AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

```
      AWS_ROLE_ARN:
 arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:            /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

Dois conjuntos de resultados duplicados são retornados porque o Pod contém dois contêineres. Os dois contêineres têm os mesmos valores.

Se o seu Pod estiver utilizando o endpoint da Região da AWS, a linha a seguir também será retornada na saída anterior.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Etapa 3: Remover a política CNI do perfil do IAM do nó

Se a função do IAM de nó do Amazon EKS atualmente tem a `AmazonEKS_CNI_Policy`política do IAM (`IPv4`) ou uma políticaIPv6anexada a ela, e você criou uma função do IAM separada, anexou a política a ela e a atribuiu à conta de serviço `aws-node`, então recomendamos que você remova a política da sua função de nó com o comando Kubernetes correspondente à AWS CLI família de IP do seu cluster. Substitua *AmazonEKSNodeRole* pelo nome da função do nó.

- `IPv4`

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn
 arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

- `IPv6`

  Substitua *111122223333* pelo ID da conta e *AmazonEKS_CNI_IPv6_Policy* pelo nome da política de `IPv6`.

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn
 arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

Criar uma política do IAM para clusters que usam a família **`IPv6`**

Se você tiver criado um cluster que usa a família `IPv6` e esse cluster tiver a versão `1.10.1` ou superior do complemento Amazon VPC CNI plugin for Kubernetes configurado, será necessário criar

uma política do IAM que você possa atribuir a um perfil do IAM. Se você tiver um cluster que não foi configurado com a família `IPv6` no momento da criação, será necessário criar um novo cluster para usar `IPv6`. Para obter mais informações sobre como usar o `IPv6` com o seu cluster, consulte Endereços `IPv6` para clusters, Pods e services.

1. Copie o texto a seguir e salve-o em um arquivo chamado *`vpc-cni-ipv6-policy`*`.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. Crie a política do IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

Escolher de casos de uso de rede de Pod

O Amazon VPC CNI plugin for Kubernetes fornece conectividade de rede para Pods. A tabela a seguir ajuda a entender quais casos de uso de redes podem ser usados juntos e os recursos e as configurações do Amazon VPC CNI plugin for Kubernetes que é possível usar com diferentes tipos de nós do Amazon EKS. Todas as informações na tabela se aplicam apenas aos nós Linux do `IPv4`.

Tipo de nó do Amazon EKS	Amazon EC2			Fargate
Caso de uso	Endereços IP individuais atribuídos à interface de rede	Prefixos IP atribuídos à interface de rede	Grupos de segurança do Pods	
Rede personalizada para pods – Atribuir endereços IP de uma sub-rede diferente da sub-rede do nó	Sim	Sim	Sim	Sim (sub-redes controladas pelo perfil do Fargate)
SNAT para Pods	Sim (o padrão é `false`)	Sim (o padrão é `false`)	Sim (`true` apenas)	Sim (`true` apenas)
Capacidades				
escopo dos grupos de segurança	Nó	Nó	Pod (se você definiu `POD_SECURITY_GROUP_ENFORCING_MODE = standard` e `AWS_VPC_K8S_CNI_EXTERNALSNA`	Pod

Tipo de nó do Amazon EKS	Amazon EC2			Fargate
Caso de uso	Endereços IP individuais atribuídos à interface de rede	Prefixos IP atribuídos à interface de rede	Grupos de segurança do Pods	
			`T =false`, o tráfego destinado a endpoints fora da VPC usa os grupos de segurança do nó, e não os grupos de segurança do Pod's)	
Tipos de sub-rede da Amazon VPC	Públicos e privados	Públicos e privados	Privado apenas	Privado apenas
Política de rede (VPC CNI)	Compatível	Compatível	Compatível Apenas com a versão `1.14.0` ou posterior do plug-in Amazon VPC CNI	Não suportado
Densidade de pods por nó	Médio	Alta	Baixo	Um
Hora do lançamento do pod	Melhor	O melhor	Bom	Moderada

Tipo de nó do Amazon EKS	Amazon EC2			Fargate
Caso de uso	Endereços IP individuais atribuídos à interface de rede	Prefixos IP atribuídos à interface de rede	Grupos de segurança do Pods	
Configurações do plug-in Amazon VPC CNI (para obter mais informações sobre cada configuração, consulte amazon-vpc-cni-k8s no GitHub)				
`WARM_ENI_TARGET`	Sim	Não aplicável	Não aplicável	Não aplicável
`WARM_IP_TARGET`	Sim	Sim	Não aplicável	Não aplicável
`MINIMUM_IP_TARGET`	Sim	Sim	Não aplicável	Não aplicável
`WARM_PREFIX_TARGET`	Não aplicável	Sim	Não aplicável	Não aplicável

Note

- Não é possível usar o `IPv6` com redes personalizadas.
- Os endereços `IPv6` não são traduzidos. Consequentemente, a SNAT não se aplica.
- O fluxo de tráfego entrando e saindo dos Pods e os grupos de segurança associados não está sujeito à política de rede do Calico e está limitado apenas à imposição do grupo de segurança da Amazon VPC.
- Se você usar a aplicação da política de rede Calico, recomendamos definir a variável de ambiente `ANNOTATE_POD_IP` como `true` para evitar um problema conhecido do Kubernetes. Para usar esse recurso, você deve adicionar a permissão `patch` para pods ao ClusterRole do `aws-node`. Observe que adicionar permissões de aplicação de patches ao DaemonSet do `aws-node` aumenta o escopo de segurança do plug-in. Para obter

mais informações, consulte ANNOTATE_POD_IP no repositório do plug-in CNI da VPC no GitHub.

- Prefixos IP e endereços IP estão associados a interfaces de rede elásticas padrão do Amazon EC2. Os pods que exigem grupos de segurança específicos recebem o endereço IP principal de uma interface de rede de filial. Você pode misturar Pods que recebem endereços IP ou endereços IP de prefixos IP com Pods que recebem interfaces de rede de ramificação no mesmo nó.

Nós do Windows

Cada nó oferece suporte somente a uma interface de rede. Você pode usar endereços `IPv4` secundários e prefixos `IPv4`. Por padrão, o número de endereços `IPv4` disponíveis no nó é igual ao número de endereços `IPv4` secundários que é possível atribuir a cada interface de rede elástica, menos um. No entanto, é possível aumentar os endereços `IPv4` disponíveis e a densidade de Pod no nó por meio da habilitação de prefixos IP. Para ter mais informações, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2.

As políticas de rede Calico são aceitas pelo Windows. Não é possível usar grupos de segurança para Pods ou redes personalizadas no Windows.

Endereços **`IPv6`** para clusters, Pods e services

Por padrão, o Kubernetes atribui endereços `IPv4` aos Pods e services. Em vez de atribuir endereços `IPv4` aos seus Pods e services, é possível configurar o seu cluster para atribuir endereços `IPv6` a eles. O Amazon EKS não é compatível com pilhas duplas de Pods ou services, embora o Kubernetes seja na versão`1.23` e posterior. Consequentemente, você não pode atribuir endereços `IPv4` e `IPv6` aos seus Pods e services.

Você seleciona qual família de IPs deseja usar para o seu cluster ao criá-lo. Não será possível alterar a família depois de criar o cluster.

Considerações sobre o uso da família **`IPv6`** para o cluster

- Você deve criar um novo cluster e especificar que deseja usar a família `IPv6` para esse cluster. Você não pode habilitar a família do `IPv6` para um cluster atualizado de uma versão anterior. Para obter instruções sobre como criar um novo cluster, consulte Criar um cluster do Amazon EKS..
- A versão do complemento CNI da Amazon VPC que você implanta em seu cluster deve ser a versão `1.10.1` ou superior. Essa versão ou uma posterior é implantada por padrão. Depois de

implantar o complemento, você não poderá fazer downgrade do seu complemento CNI da Amazon VPC para uma versão inferior a `1.10.1` sem remover primeiro todos os nós em todos os grupos de nós do seu cluster.

- Os Pods e services do Windows não são compatíveis.
- Se você usar nós do Amazon EC2, deverá configurar o complemento CNI da Amazon VPC com delegação de prefixo IP e `IPv6`. Se você escolher a família `IPv6` ao criar o seu cluster, a versão `1.10.1` do complemento elegerá tal configuração como padrão. Esse é o caso de complementos autogerenciados ou do Amazon EKS. Para obter mais informações sobre a delegação de prefixos IP, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2.
- Ao criar um cluster, a VPC e as sub-redes que você especificar deverão ter um bloco CIDR `IPv6` atribuído a elas. Elas também precisam ter um bloco CIDR `IPv4` atribuído a elas. Isso ocorre porque, ainda que você só queira usar o `IPv6`, uma VPC ainda precisa de um bloco CIDR `IPv4` para funcionar. Para obter mais informações, consulte Associar um bloco CIDR `IPv6` à sua VPC, no Guia do usuário da Amazon VPC.
- Ao criar o seu cluster e nós, você deve especificar sub-redes configuradas para atribuir automaticamente endereços `IPv6`. Caso contrário, você não poderá implantar o seu cluster e nós. Por padrão, essa configuração fica desativada. Para obter mais informações, consulte Modificar o atributo de endereçamento `IPv6` para a sua sub-rede no Guia do usuário da Amazon VPC.
- As tabelas de rotas atribuídas às suas sub-redes devem ter rotas para endereços `IPv6`. Para obter mais informações, consulte Migrar para `IPv6` no Guia do usuário da Amazon VPC.
- Os seus grupos de segurança devem permitir endereços `IPv6`. Para obter mais informações, consulte Migrar para `IPv6` no Guia do usuário da Amazon VPC.
- Você só pode usar o `IPv6` com nós do Amazon EC2 baseados em AWS Nitro ou do Fargate.
- É possível usar `IPv6` com Grupos de segurança do Pods com nós do Amazon EC2 e nós do Fargate.
- Se você usou redes personalizadas anteriormente para aliviar o esgotamento de endereços IP, poderá substituí-las por `IPv6`. Não é possível usar redes personalizadas com `IPv6`. Se você usar redes personalizadas para isolar redes, talvez seja necessário continuar usando redes personalizadas e a família `IPv4` para os seus clusters.
- Não é possível usar `IPv6` com AWS Outposts.
- Serviços do Kubernetes só são atribuídos a um endereço IPv6. Isso não ocorre com um endereço IPv4.
- Os pods recebem um endereço IPv6 e um endereço IPv4 local do host. O endereço IPv4 local do host é atribuído usando um plug-in CNI local do host encadeado com o VPC CNI, e o endereço

não é informado ao ambiente de gerenciamento do Kubernetes. Ele só é usado quando um pod precisa se comunicar com recursos IPv4 externos em outra Amazon VPC ou na internet. O endereço IPv4 local do host é SNATed (pelo VPC CNI) para o endereço IPv4 primário da ENI primária do nó de processamento.

- Pods e services recebem apenas um endereço `IPv6`. Isso não ocorre com um endereço `IPv4`. Como os Pods podem se comunicar com endpoints `IPv4` por meio do NAT na própria instância, DNS64 e NAT64 não são necessários. Se o tráfego precisar de um endereço IP público, ele será o endereço de rede de origem traduzido para um IP público.
- O endereço `IPv6` de origem de um Pod não é o endereço de rede de origem traduzido para o endereço `IPv6` do nó em um momento de comunicação externa à VPC. Ele é roteado usando um gateway da Internet ou um gateway da Internet somente de saída.
- Todos os nós são atribuídos a endereços `IPv4` e `IPv6`.
- O Armazene aplicações de alta performance com o FSx para Lustre não é compatível.
- É possível usar a versão `2.3.1` ou superior do AWS Load Balancer Controller para balancear a carga de tráfego de aplicações ou redes para `IPv6` Pods no modo IP, mas não no modo de instância. Para obter mais informações, consulte O que é o AWS Load Balancer Controller?.
- Você deve anexar uma política do IAM `IPv6` ao seu perfil do IAM do CNI ou IAM do nó. Entre os dois, recomendamos que você a anexe a uma função do IAM do CNI. Para ter mais informações, consulte Criar uma política do IAM para clusters que usam a família `IPv6` e Etapa 1: Criar o perfil do IAM Amazon VPC CNI plugin for Kubernetes.
- Cada Pod do Fargate recebe um endereço `IPv6` do CIDR especificado para a sub-rede em que está implantado. A unidade de hardware subjacente que executa Pods do Fargate obtém um endereço `IPv4` e `IPv6` exclusivo dos CIDRs atribuídos à sub-rede em que a unidade de hardware está implantada.
- Recomendamos que você faça uma avaliação completa de suas aplicações, complementos do Amazon EKS e produtos da AWS com os quais você se integra antes de implantar clusters do `IPv6`. Isso ocorre para garantir que tudo funcione conforme o esperado com o `IPv6`.
- O uso do endpoint `IPv6` do Serviço de metadados de instância do Amazon EC2 não é compatível com o Amazon EKS.
- Ao criar um grupo autogerenciado de nós em um cluster que use `IPv6`, os dados do usuário devem incluir os seguintes `BootstrapArguments` para o arquivo `bootstrap.sh` que é executado na inicialização do nó. Substitua *`your-cidr`* pelo intervalo CIDR `IPv6` da VPC do seu cluster.

```
--ip-family ipv6 --service-ipv6-cidr your-cidr
```

Se você não souber o intervalo CIDR IPv6 para seu cluster, será possível visualizá-lo com o seguinte comando (requer o AWS CLI versão 2.4.9 ou posterior).

```
aws eks describe-cluster --name my-cluster --query
 cluster.kubernetesNetworkConfig.serviceIpv6Cidr --output text
```

Implantar um cluster **IPv6** e nós gerenciados do Amazon Linux

Neste tutorial, você implanta uma Amazon VPC IPv6, um cluster do Amazon EKS com a família IPv6 e um grupo de nós gerenciados com nós do Amazon Linux do Amazon EC2. Não é possível implantar nós do Amazon EC2 Windows em um cluster IPv6. Você também pode implantar nós do Fargate em seu cluster. Para simplificar, tais instruções não são fornecidas neste tópico.

Antes de criar um cluster para uso em produção, recomendamos que você se familiarize com todas as configurações e implante um cluster com as configurações que atendam aos seus requisitos. Para obter mais informações, consulte Criar um cluster do Amazon EKS., Simplificar o ciclo de vida dos nós com grupos de nós gerenciados e as considerações deste tópico. Só é possível habilitar algumas configurações durante a criação do seu cluster.

Pré-requisitos

Antes de iniciar este tutorial, você deve instalar e configurar as ferramentas a seguir e os recursos necessários para criar e gerenciar um cluster do Amazon EKS.

- A ferramenta da linha de comando kubectl está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a 1.29, você poderá usar o kubectl versão 1.28, 1.29 ou 1.30 com ele. Para instalar ou atualizar o kubectl, consulte Configurar o kubectl e o eksctl.
- A entidade principal de segurança do IAM que você está usando deve ter permissões para trabalhar com perfis do IAM do Amazon EKS, funções vinculadas ao serviço, AWS CloudFormation, uma VPC e recursos relacionados. Para obter mais informações, consulte Ações, recursos e chaves de condição do Amazon Elastic Kubernetes Service e Usar funções vinculadas a serviço no Guia do usuário do IAM.

Os procedimentos são fornecidos para criar os recursos com `eksctl` ou a AWS CLI. Você também pode implantar os recursos usando o AWS Management Console. Para simplificar, tais instruções não são fornecidas neste tópico.

eksctl

Pré-requisito

O `eksctl` versão `0.187.0` ou posterior instalada no computador. Para instalá-la ou atualizá-la, consulte Instalação na documentação do `eksctl`.

Para implantar um cluster **`IPv6`** com **`eksctl`**

1. Crie o arquivo `ipv6-cluster.yaml`. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado:

   - Substitua *`my-cluster`* por um nome de cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
   - Substitua *`region-code`* por qualquer Região da AWS compatível com o Amazon EKS. Para obter uma lista das Regiões da AWS, consulte Endpoints e cotas do Amazon EKS no guia Referência geral da AWS.
   - O valor para `version` com a versão do seu cluster. Para obter mais informações, consulte a versão do Kubernetes do Amazon EKS compatível.
   - Substitua *`my-nodegroup`* por um nome para o seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.
   - Substitua *`t3.medium`* por qualquer tipo de instância do AWS Nitro System.

```
cat >ipv6-cluster.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
```

```
  region: region-code
  version: "X.XX"

kubernetesNetworkConfig:
  ipFamily: IPv6

addons:
  - name: vpc-cni
    version: latest
  - name: coredns
    version: latest
  - name: kube-proxy
    version: latest

iam:
  withOIDC: true

managedNodeGroups:
  - name: my-nodegroup
    instanceType: t3.medium
EOF
```

2. Crie o seu cluster.

```
eksctl create cluster -f ipv6-cluster.yaml
```

A criação do cluster leva muitos minutos. Não prossiga até ver a última linha de saída, que deve parecer com a seguinte.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

3. Confirme se os Pods padrão são atribuídos a endereços IPv6.

```
kubectl get pods -n kube-system -o wide
```

Veja um exemplo de saída abaixo.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                              NODE
 NOMINATED NODE   READINESS GATES
```

```
aws-node-rslts              1/1     Running   1          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh              1/1     Running   0          5m32s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2    1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8    1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk            1/1     Running   0          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g            1/1     Running   0          5m33s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

4. Confirme se os serviços padrão são atribuídos a endereços IPv6.

```
kubectl get services -n kube-system -o wide
```

Veja um exemplo de saída abaixo.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m
 k8s-app=kube-dns
```

5. (Opcional) Implante uma aplicação de amostra ou implante o AWS Load Balancer Controller e uma aplicação de amostra para balancear a carga do tráfego da aplicação ou da rede para IPv6 Pods.

6. Após a conclusão de criação do cluster e nós para este tutorial, você deverá limpar os recursos que criou com o comando a seguir.

```
eksctl delete cluster my-cluster
```

AWS CLI

Pré-requisito

A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell. Se você usar o AWS CloudShell, talvez precise instalar a versão `2.12.3` ou posterior ou `1.27.160` ou posterior do AWS CLI, pois é possível que a versão padrão da AWS CLI instalada no AWS CloudShell seja anterior.

⚠ Important

- Você deve concluir todas as etapas deste procedimento como o mesmo usuário. Execute o seguinte comando para verificar o usuário atual:

  ```
  aws sts get-caller-identity
  ```

- Você deve concluir todas as etapas deste procedimento no mesmo shell. Várias etapas usam variáveis definidas em etapas anteriores. As etapas que usam variáveis não funcionarão corretamente se os valores das variáveis forem definidos em um shell diferente. Se você usar o AWS CloudShell para concluir o procedimento a seguir, lembre-se de que, se você não interagir com ele usando o seu teclado ou ponteiro por aproximadamente 20 a 30 minutos, a sua sessão do shell será encerrada. Os processos em execução não contam como interações.
- As instruções são redigidas para o shell bash e podem precisar de ajustes em outros shells.

Para criar o cluster com a AWS CLI

Substitua todos os *`example values`* nas etapas deste procedimento por seus próprios valores.

1. Execute os comandos a seguir para definir algumas variáveis usadas em etapas posteriores. Substitua *region-code* pela Região da AWS na qual deseja implantar os recursos. O valor pode ser qualquer Região da AWS compatível com o Amazon EKS. Para obter uma lista das Regiões da AWS, consulte Endpoints e cotas do Amazon EKS no guia Referência geral da AWS. Substitua *my-cluster* por um nome de cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster. Substitua *my-nodegroup* por um nome para o seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Substitua *111122223333* pelo ID da sua conta.

   ```
   export region_code=region-code
   export cluster_name=my-cluster
   export nodegroup_name=my-nodegroup
   export account_id=111122223333
   ```

2. Crie uma Amazon VPC com sub-redes públicas e privadas que atenda aos requisitos do Amazon EKS e IPv6.

   a. Execute o seguinte comando para definir uma variável para o seu nome de pilha do AWS CloudFormation. Você pode substituir *my-eks-ipv6-vpc* por qualquer nome que escolher.

      ```
      export vpc_stack_name=my-eks-ipv6-vpc
      ```

   b. Crie uma VPC IPv6 usando um modelo do AWS CloudFormation.

      ```
      aws cloudformation create-stack --region $region_code --stack-name
       $vpc_stack_name \
        --template-url https://s3.us-west-2.amazonaws.com/amazon-
      eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-
      subnets.yaml
      ```

      A pilha leva alguns minutos para ser criada. Execute o seguinte comando . Não prossiga para a próxima etapa até que a saída do comando seja CREATE_COMPLETE.

      ```
      aws cloudformation describe-stacks --region $region_code --stack-name
       $vpc_stack_name --query Stacks[].StackStatus --output text
      ```

c. Recupere os IDs das sub-redes públicas que foram criadas.

```
aws cloudformation describe-stacks --region $region_code --stack-name
 $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --
output text
```

Veja um exemplo de saída abaixo.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE
```

d. Habilite a opção de atribuição automática de endereços IPv6 para as sub-redes públicas que foram criadas.

```
aws ec2 modify-subnet-attribute --region $region_code --
subnet-id subnet-0a1a56c486EXAMPLE --assign-ipv6-address-on-
creation
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-099e6ca77aEXAMPLE --assign-ipv6-address-on-creation
```

e. Recupere os nomes das sub-redes e grupos de segurança criados pelo modelo na pilha implantada do AWS CloudFormation e armazene-os em variáveis para usar em uma etapa posterior.

```
security_groups=$(aws cloudformation describe-stacks --region $region_code
 --stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SecurityGroups`].OutputValue' --
output text)

public_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --
output text)

private_subnets=$(aws cloudformation describe-stacks --region $region_code
 --stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPrivate`].OutputValue' --
output text)

subnets=${public_subnets},${private_subnets}
```

3. Crie um perfil do IAM de cluster e anexe a ele a política gerenciada pelo IAM no Amazon EKS necessária. Os clusters do Kubernetes gerenciados pelo Amazon EKS fazem chamadas para outros serviços da AWS em seu nome para gerenciar os recursos que você usa com o serviço.

   a. Execute o seguinte comando para criar o arquivo `eks-cluster-role-trust-policy.json`.

   ```
   cat >eks-cluster-role-trust-policy.json <<EOF
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "eks.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   EOF
   ```

   b. Execute o seguinte comando para definir uma variável para o nome da sua função. Você pode substituir *myAmazonEKSClusterRole* por qualquer nome que escolher.

   ```
   export cluster_role_name=myAmazonEKSClusterRole
   ```

   c. Crie a função.

   ```
   aws iam create-role --role-name $cluster_role_name --assume-role-policy-
   document file://"eks-cluster-role-trust-policy.json"
   ```

   d. Recupere o ARN da função do IAM e o armazene em uma variável para uma etapa posterior.

   ```
   cluster_iam_role=$(aws iam get-role --role-name $cluster_role_name --
   query="Role.Arn" --output text)
   ```

   e. Anexe a política de IAM gerenciada pelo Amazon EKS à função.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

4. Crie o seu cluster.

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.XX \
   --role-arn $cluster_iam_role --resources-vpc-config subnetIds=
$subnets,securityGroupIds=$security_groups \
   --kubernetes-network-config ipFamily=ipv6
```

-
> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte Insufficient capacity (Capacidade insuficiente).

A criação do cluster demora alguns minutos. Execute o seguinte comando . Não prossiga para a próxima etapa até que a saída do comando esteja `ACTIVE`.

```
aws eks describe-cluster --region $region_code --name $cluster_name --query
 cluster.status
```

5. Crie ou atualize um arquivo `kubeconfig` para o seu cluster para que você possa se comunicar com ele.

```
aws eks update-kubeconfig --region $region_code --name $cluster_name
```

Por padrão, um arquivo `~/.kube` é criado em `config` ou o novo cluster é adicionado a um arquivo `config` existente no `~/.kube`.

6. Crie uma função do IAM para o nó.

   a. Execute o seguinte comando para criar o arquivo `vpc-cni-ipv6-policy.json`.

```
cat >vpc-cni-ipv6-policy <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
EOF
```

b. Crie a política do IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

c. Execute o seguinte comando para criar o arquivo `node-role-trust-relationship.json`.

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

d. Execute o seguinte comando para definir uma variável para o nome da sua função. Você pode substituir *AmazonEKSNodeRole* por qualquer nome que escolher.

```
export node_role_name=AmazonEKSNodeRole
```

e. Crie o perfil do IAM.

```
aws iam create-role --role-name $node_role_name --assume-role-policy-
document file://"node-role-trust-relationship.json"
```

f. Anexe a política do IAM à função do IAM.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy \
    --role-name $node_role_name
```

> **Important**
>
> Para simplificar este tutorial, a política é anexada a esta função do IAM. No entanto, em um cluster de produção, recomendamos que você anexe a política a uma função do IAM separada. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).

g. Anexe duas políticas do IAM gerenciadas necessárias à função do IAM.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly \
```

```
  --role-name $node_role_name
```

h. Recupere o ARN da função do IAM e o armazene em uma variável para uma etapa posterior.

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --
query="Role.Arn" --output text)
```

7. Crie um grupo de nós gerenciado.

a. Exiba os IDs das sub-redes que você criou em uma etapa anterior.

```
echo $subnets
```

Veja um exemplo de saída abaixo.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE,subnet-
0377963d69EXAMPLE,subnet-0c05f819d5EXAMPLE
```

b. Crie o grupo de nós. Substitua *0a1a56c486EXAMPLE*, *099e6ca77aEXAMPLE*, *0377963d69EXAMPLE* e *0c05f819d5EXAMPLE* pelos valores retornados na entrada da etapa anterior. Certifique-se de remover as vírgulas entre IDs de sub-redes da saída anterior no comando a seguir. Você pode substituir *t3.medium* por qualquer tipo de instância do AWS Nitro System.

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name
 --nodegroup-name $nodegroup_name \
    --subnets subnet-0a1a56c486EXAMPLE subnet-099e6ca77aEXAMPLE
 subnet-0377963d69EXAMPLE subnet-0c05f819d5EXAMPLE \
    --instance-types t3.medium --node-role $node_iam_role
```

A criação do grupo de nós demora alguns minutos. Execute o seguinte comando . Não prossiga para a próxima etapa até que a saída recebida esteja `ACTIVE`.

```
aws eks describe-nodegroup --region $region_code --cluster-name
 $cluster_name --nodegroup-name $nodegroup_name \
    --query nodegroup.status --output text
```

8. Confirme se os Pods padrão são atribuídos a endereços `IPv6` na coluna `IP`.

```
kubectl get pods -n kube-system -o wide
```

Veja um exemplo de saída abaixo.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                             NODE
 NOMINATED NODE   READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

9. Confirme se os serviços padrão são atribuídos a endereços IPv6 na coluna IP.

```
kubectl get services -n kube-system -o wide
```

Veja um exemplo de saída abaixo.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m
 k8s-app=kube-dns
```

10. (Opcional) Implante uma aplicação de amostra ou implante o AWS Load Balancer Controller e uma aplicação de amostra para balancear a carga do tráfego da aplicação ou da rede para IPv6 Pods.

11. Após a conclusão de criação do cluster e nós para este tutorial, você deverá limpar os recursos que criou com os comandos a seguir. Certifique-se de que você não esteja usando nenhum dos recursos fora deste tutorial antes de excluí-los.

    a. Se você estiver realizando esta etapa em um shell diferente daquele em que realizou as etapas anteriores, defina os valores de todas as variáveis usadas nas etapas anteriores, substituindo os *example values* pelos valores especificados ao realizar as etapas anteriores. Se você estiver concluindo esta etapa no mesmo shell em que concluiu as etapas anteriores, pule para a próxima etapa.

    ```
    export region_code=region-code
    export vpc_stack_name=my-eks-ipv6-vpc
    export cluster_name=my-cluster
    export nodegroup_name=my-nodegroup
    export account_id=111122223333
    export node_role_name=AmazonEKSNodeRole
    export cluster_role_name=myAmazonEKSClusterRole
    ```

    b. Exclua o seu grupo de nós.

    ```
    aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name
     --nodegroup-name $nodegroup_name
    ```

    A exclusão demora alguns minutos. Execute o seguinte comando . Não prossiga para a próxima etapa se receber alguma saída.

    ```
    aws eks list-nodegroups --region $region_code --cluster-name $cluster_name
     --query nodegroups --output text
    ```

    c. Excluir o cluster.

    ```
    aws eks delete-cluster --region $region_code --name $cluster_name
    ```

    Demora alguns minutos até que o cluster seja excluído. Antes de continuar, verifique se o cluster foi excluído com o seguinte comando.

    ```
    aws eks describe-cluster --region $region_code --name $cluster_name
    ```

    Não prossiga para a próxima etapa até que a saída seja semelhante à seguinte saída.

```
An error occurred (ResourceNotFoundException) when calling the
 DescribeCluster operation: No cluster found for name: my-cluster.
```

d. Exclua os recursos do IAM que você criou. Substitua *AmazonEKS_CNI_IPv6_Policy* pelo nome que você escolheu, se ele for diferente do usado nas etapas anteriores.

```
aws iam detach-role-policy --role-name $cluster_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-policy --policy-arn arn:aws:iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-role --role-name $cluster_role_name
aws iam delete-role --role-name $node_role_name
```

e. Exclua a pilha do AWS CloudFormation que criou a VPC.

```
aws cloudformation delete-stack --region $region_code --stack-name
 $vpc_stack_name
```

SNAT para Pods

Se você implantou seu cluster utilizando a família `IPv6`, as informações neste tópico não serão aplicáveis a ele, pois endereços `IPv6` não são convertidos na rede. Para obter mais informações sobre como usar o `IPv6` com o seu cluster, consulte Endereços `IPv6` para clusters, Pods e services.

Por padrão, cada Pod no cluster recebe um endereço `IPv4` privado de um bloco de Encaminhamento Entre Domínios Sem Classificação (CIDR) associado à VPC na qual o Pod está implantado. Pods na mesma VPC se comunicam uns com os outros utilizando esses endereços IP privados como endpoints. Quando um Pod se comunica com qualquer endereço `IPv4` que não está dentro de um bloco CIDR associado à VPC, o plug-in CNI da Amazon VPC (para Linux ou Windows) converte endereço `IPv4` do Pod's no endereço `IPv4` privado da interface de rede elástica primária do nó no qual o Pod está em execução, por padrão [*].

> **Note**
>
> Para nós Windows, existem detalhes adicionais a serem considerados. Por padrão, o plug-in CNI da VPC CNI para Windows é definido com uma configuração de rede em que o tráfego para um destino dentro da mesma VPC é excluído para SNAT. Isso significa que a comunicação da VPC interna tem a SNAT desabilitada e que o endereço IP alocado a um Pod é roteável dentro da VPC. Porém, o tráfego para um destino fora da VPC tem o IP de origem conectado Pod ao endereço IP primária da ENI da instância. Essa configuração padrão para o Windows garante que o pod possa acessar redes fora da sua VPC da mesma forma que a instância do host.

Devido a este comportamento:

- Os Pods poderão se comunicar com os recursos da Internet somente se o nó em que estão sendo executados tiver um endereço IP público ou elástico atribuído a ele e se estiver em uma sub-rede pública. A tabela de rotas associada a uma sub-rede pública tem uma rota para um gateway da Internet. Convém implantar nós em sub-redes privadas, sempre que possível.
- Para versões de plug-in anteriores a `1.8.0`, os recursos que estão em redes ou VPCs conectadas à sua VPC do cluster usando emparelhamento de VPC, uma VPC em trânsito ou o AWS Direct Connect não poderão iniciar a comunicação com os Pods por interfaces de rede elástica secundárias. Seus Pods podem iniciar a comunicação com esses recursos e receber respostas deles.

Se qualquer uma das declarações a seguir for verdadeira em seu ambiente, altere a configuração padrão com o comando a seguir.

- Você tem recursos em redes ou VPCs que estão conectados à sua VPC do cluster usando emparelhamento de VPC, uma VPC em trânsito ou o AWS Direct Connect que precisam iniciar a comunicação com seus Pods usando um endereço `IPv4` e sua versão de plug-in deve ser anterior à `1.8.0`.
- Os Pods estão em uma sub-rede privada e precisam se comunicar com a saída de Internet. A sub-rede tem uma rota para um gateway NAT.

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

> **Note**
>
> As variáveis de configuração CNI `AWS_VPC_K8S_CNI_EXTERNALSNAT` e `AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS` não são aplicáveis a nós Windows. A desativação do SNAT não tem suporte no Windows. Quanto à exclusão de uma lista de CIDRs `IPv4` do SNAT, você pode definir isso especificando o parâmetro `ExcludedSnatCIDRs` no script de bootstrap Windows. Para obter mais informações sobre o uso desse parâmetro, consulte Parâmetros de configuração do script de bootstrap.

*Se a especificação de Pod's contiver `hostNetwork=true` (o padrão é `false`), seu endereço IP não será convertido em um endereço diferente. Este é o caso dos Pods Kubernetes de `kube-proxy` e do Amazon VPC CNI plugin for , que são executados no seu cluster por padrão. Para esses Pods, o endereço IP é o mesmo que o endereço IP primário do nó e, portanto, o endereço IP do Pod's não é convertido. Para obter mais informações sobre a configuração de `hostNetwork` do Pod's, consulte PodSpec v1 core na referência de APIs do Kubernetes.

Configure seu cluster para as políticas de rede do Kubernetes

Por padrão, não há restrições no Kubernetes para endereços IP, portas ou conexões entre os Pods do seu cluster ou entre os Pods e os recursos de qualquer outra rede. Você pode usar a política de rede do Kubernetes para restringir o tráfego de rede que entra e que sai dos Pods. Para obter mais informações, consulte Política de segurança de pods na documentação do Kubernetes.

Se você tiver a versão `1.13` ou anterior do Amazon VPC CNI plugin for Kubernetes em seu cluster, precisará implementar uma solução de terceiros para aplicar as políticas de rede do Kubernetes ao cluster. A versão `1.14`, ou posterior, do plug-in pode implementar políticas de rede, para que você não precise usar uma solução de terceiros. Neste tópico, você aprende a configurar o cluster para usar a política de rede do Kubernetes no cluster sem usar um complemento de terceiros.

As políticas de rede no Amazon VPC CNI plugin for Kubernetes são compatíveis com as configurações a seguir.

- Clusters do Amazon EKS da versão `1.25` e posteriores.
- Versão 1.14 ou posterior do Amazon VPC CNI plugin for Kubernetes no cluster.
- Cluster configurado para endereços `IPv4` ou `IPv6`.
- Você pode usar políticas de rede com grupos de segurança para os Pods. Com as políticas de rede, você pode controlar toda a comunicação dentro do cluster. Com os grupos de segurança

para os Pods, você pode controlar o acesso aos Serviços da AWS das aplicações dentro de um Pod.

- Você pode usar políticas de rede com rede personalizada e delegação de prefixo.

Considerações

- Ao aplicar as políticas de rede do Amazon VPC CNI plugin for Kubernetes no cluster com o Amazon VPC CNI plugin for Kubernetes, você só pode aplicar as políticas aos nós do Linux do Amazon EC2 . Você não pode aplicar as políticas aos nós do Fargate ou do Windows.
- Se o cluster estiver usando atualmente uma solução de terceiros para gerenciar as políticas de rede do Kubernetes, você poderá usar essas mesmas políticas com o Amazon VPC CNI plugin for Kubernetes. Porém, você deve remover a solução existente para que ela não gerencie as mesmas políticas.
- Você pode aplicar várias políticas de rede ao mesmo Pod. Quando duas ou mais políticas que selecionam o mesmo Pod estão configuradas, todas as políticas são aplicadas ao Pod.
- O número máximo de combinações exclusivas de portas para cada protocolo em cada seletor `ingress:` ou `egress:` em uma política de rede é 24.
- Para qualquer serviço do Kubernetes, a porta de serviço deve ser a mesma que porta de contêiner. Se você estiver usando portas nomeadas, use o mesmo nome na especificação do serviço.
- Aplicação de políticas na inicialização do Pod

  O Amazon VPC CNI plugin for Kubernetes configura as políticas de rede para os pods em paralelo ao provisionamento de pods. Até que todas as políticas estejam configuradas para o novo pod, os contêineres no novo pod serão iniciados com uma política de permissão padrão. Isso é denominado modo padrão. Uma política de permissão padrão significa que todo o tráfego de entrada e de saída é permitido de e para os novos pods. Por exemplo, os pods não terão nenhuma regra de firewall aplicada (todo o tráfego é permitido) até que o novo pod seja atualizado com as políticas ativas.

  É possível alterar essa política de rede padrão ao definir a variável de ambiente `NETWORK_POLICY_ENFORCING_MODE` como `strict` no contêiner `aws-node` do objeto `DaemonSet` do plug-in CNI da VPC.

```
env:
  - name: NETWORK_POLICY_ENFORCING_MODE
```

```
    value: "strict"
```

Com a variável `NETWORK_POLICY_ENFORCING_MODE` definida como `strict`, os pods que usam o plug-in CNI da VPC são iniciados com uma política de negação padrão e, em seguida, as políticas são configuradas. Isso é denominado modo estrito. No modo estrito, você deve ter uma política de rede para cada endpoint que os pods precisam acessar no cluster. Observe que esse requisito se aplica aos pods CoreDNS. A política de negação padrão não está configurada para pods com sistemas de redes de host.

- O atributo de política de rede cria e exige uma definição de atributos personalizados (CRD) de `PolicyEndpoint` denominada `policyendpoints.networking.k8s.aws`. Os objetos de `PolicyEndpoint` do atributo personalizado são gerenciados pelo Amazon EKS. Você não deve modificar nem excluir esses recursos.
- Se você executar pods que usem as credenciais do IAM do perfil da instância ou se conectar ao IMDS do EC2, tenha o cuidado de verificar as políticas de rede que bloqueariam o acesso ao IMDS do EC2. Pode ser necessário adicionar uma política de rede para permitir o acesso ao IMDS do EC2. Para obter mais informações, consulte Instance metadata and user data(Metadados da instância e dados do usuário) no manual do usuário do Amazon EC2.

  Pods que usam perfis do IAM para contas de serviço não acessam o IDMS do EC2.
- O Amazon VPC CNI plugin for Kubernetes não aplica políticas de rede a interfaces de rede adicionais para cada pod, somente à interface primária para cada pod (`eth0`). Isso afeta as seguintes arquiteturas:
  - Pods `IPv6` com a variável `ENABLE_V4_EGRESS` definida como `true`. Essa variável permite que o recurso de saída `IPv4` conecte os pods IPv6 a endpoints `IPv4`, como aqueles fora do cluster. O funcionamento do recurso de saída `IPv4` se dá com a criação de uma interface de rede adicional com um endereço IPv4 de loopback local.
  - Ao usar plug-ins de rede encadeados, como Multus. Como esses plug-ins adicionam interfaces de rede a cada pod, as políticas de rede não são aplicadas aos plug-ins de rede encadeados.
- O recurso de política de rede usa a porta `8162` no nó para métricas por padrão. Além disso, o recurso usava uma porta `8163` para sondas de saúde. Se você executar outro aplicativo nos nós ou dentro dos pods que precisa usar essas portas, o aplicativo não será executado. Na versão VPC CNI `v1.14.1` ou posterior, você pode alterar a porta dessas portas nos seguintes locais:

  AWS Management Console

  1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

2. No painel de navegação à esquerda, selecione Clusters e o nome do cluster para o qual você deseja configurar o complemento Amazon VPC CNI.
3. Escolha a guia Add-ons (Complementos).
4. Selecione a caixa no canto superior direito da caixa do complemento e depois escolha Edit (Editar).
5. Na página Configure ***name of addon*** (Configurar nome do complemento):

   a. Selecione a versão `v1.14.0-eksbuild.3` ou posterior na lista suspensa Versão.

   b. Expanda Definições de configuração opcionais.

   c. Insira a chave JSON `"enableNetworkPolicy":` e o valor `"true"` em Valores da configuração. O texto resultante deve ser um objeto JSON válido. Se esse par de chave e valor for o único dado na caixa de texto, coloque-o entre colchetes `{}`.

      O seguinte exemplo tem a funcionalidade de política de rede ativada e as métricas e investigações de integridade estão configuradas com os números de porta padrão:

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "healthProbeBindAddr": "8163",
        "metricsBindAddr": "8162"
    }
}
```

Helm

Se você instalou o Amazon VPC CNI plugin for Kubernetes por meio do `helm`, você pode atualizar a configuração para alterar as portas.

- Execute o seguinte comando para alterar as portas. Defina o número da porta no valor da chave `nodeAgent.metricsBindAddr` ou da chave `nodeAgent.healthProbeBindAddr`, respectivamente.

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
 nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. Abra o `aws-node` `DaemonSet` no editor.

```
kubectl edit daemonset -n kube-system aws-node
```

2. Substitua os números da porta nos argumentos do comando a seguir no contêiner `args:` no manifesto do daemonset `aws-network-policy-agent` do CNI do `aws-node` VPC.

```
    - args:
            - --metrics-bind-addr=:8162
            - --health-probe-bind-addr=:8163
```

Pré-requisitos

- Versão mínima do cluster

  Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS. O cluster deve ser da versão `1.25` ou posterior do Kubernetes. O cluster deve estar executando uma das versões do Kubernetes e versões da plataforma listadas na tabela a seguir. Observe que qualquer Kubernetes e da plataforma posteriores às versões listadas. Para verificar a versão atual do Kubernetes substitua *`my-cluster`* no comando a seguir pelo nome do cluster e execute o comando modificado: .

```
aws eks describe-cluster
              --name my-cluster --query cluster.version --output
              text
```

Versão do Kubernetes	Versão da plataforma
`1.27.4`	`eks.5`
`1.26.7`	`eks.6`
`1.25.12`	`eks.7`

- Versão mínima do VPC CNI

Versão `1.14` ou posterior do Amazon VPC CNI plugin for Kubernetes no cluster. É possível verificar qual é a sua versão atual com o comando a seguir.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Se a versão for anterior à `1.14`, consulte Atualizar o complemento do Amazon EKS para atualizar para a versão `1.14` ou posterior.

- Versão mínima do kernel do Linux

  Os nós devem ter a versão `5.10` ou posterior do kernel do Linux. Você pode verificar qual é a sua versão atual com `uname -r`. Se você usa as versões mais recentes do Amazon Linux otimizado para Amazon EKS, das AMIs do Amazon Linux otimizadas para Amazon EKS e as AMIs do Bottlerocket, elas já têm a versão do kernel exigida.

  A AMI do Amazon Linux acelerada otimizada para Amazon EKS versão `v20231116` ou posterior possui o kernel versão `5.10`.

Para configurar o cluster para usar as políticas de rede do Kubernetes

1. Monte o sistema de arquivos BPF

   > **Note**
   >
   > Se o cluster for da versão `1.27` ou posterior, você poderá pular essa etapa, pois todas as AMIs do Amazon Linux e Bottlerocket otimizadas para o Amazon EKS `1.27` ou posterior já têm esse atributo.
   > Para todas as outras versões do cluster, se você atualizar o Amazon Linux otimizado para o Amazon EKS para a versão `v20230703` ou posterior, ou se atualizar a AMI do Bottlerocket para a versão `v1.0.2` ou posterior, poderá pular essa etapa.

   a. Monte o sistema de arquivos Berkeley Packet Filter (BPF) em cada um dos nós.

      ```
      sudo mount -t bpf bpffs /sys/fs/bpf
      ```

b. Em seguida, adicione o mesmo comando aos dados do usuário no modelo de inicialização para os grupos do Amazon EC2 Auto Scaling.

2. Habilitar a política de rede no VPC CNI

a. Veja qual tipo de complemento está atualmente instalado no cluster. Dependendo da ferramenta com a qual você criou o cluster, talvez você não tenha o tipo de complemento do Amazon EKS instalado em seu cluster atualmente. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

Se um número de versão for retornado, você tem o tipo de complemento do Amazon EKS instalado no cluster, e não precisa completar as etapas restantes deste procedimento. Se um erro for retornado, você não tem o tipo de complemento do Amazon EKS instalado no cluster.

b. • Complemento do Amazon EKS

AWS Management Console

a. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

b. No painel de navegação à esquerda, selecione Clusters e o nome do cluster para o qual você deseja configurar o complemento Amazon VPC CNI.

c. Escolha a guia Add-ons (Complementos).

d. Selecione a caixa no canto superior direito da caixa do complemento e depois escolha Edit (Editar).

e. Na página Configure ***name of addon*** (Configurar nome do complemento):

i. Selecione a versão `v1.14.0-eksbuild.3` ou posterior na lista suspensa Versão.

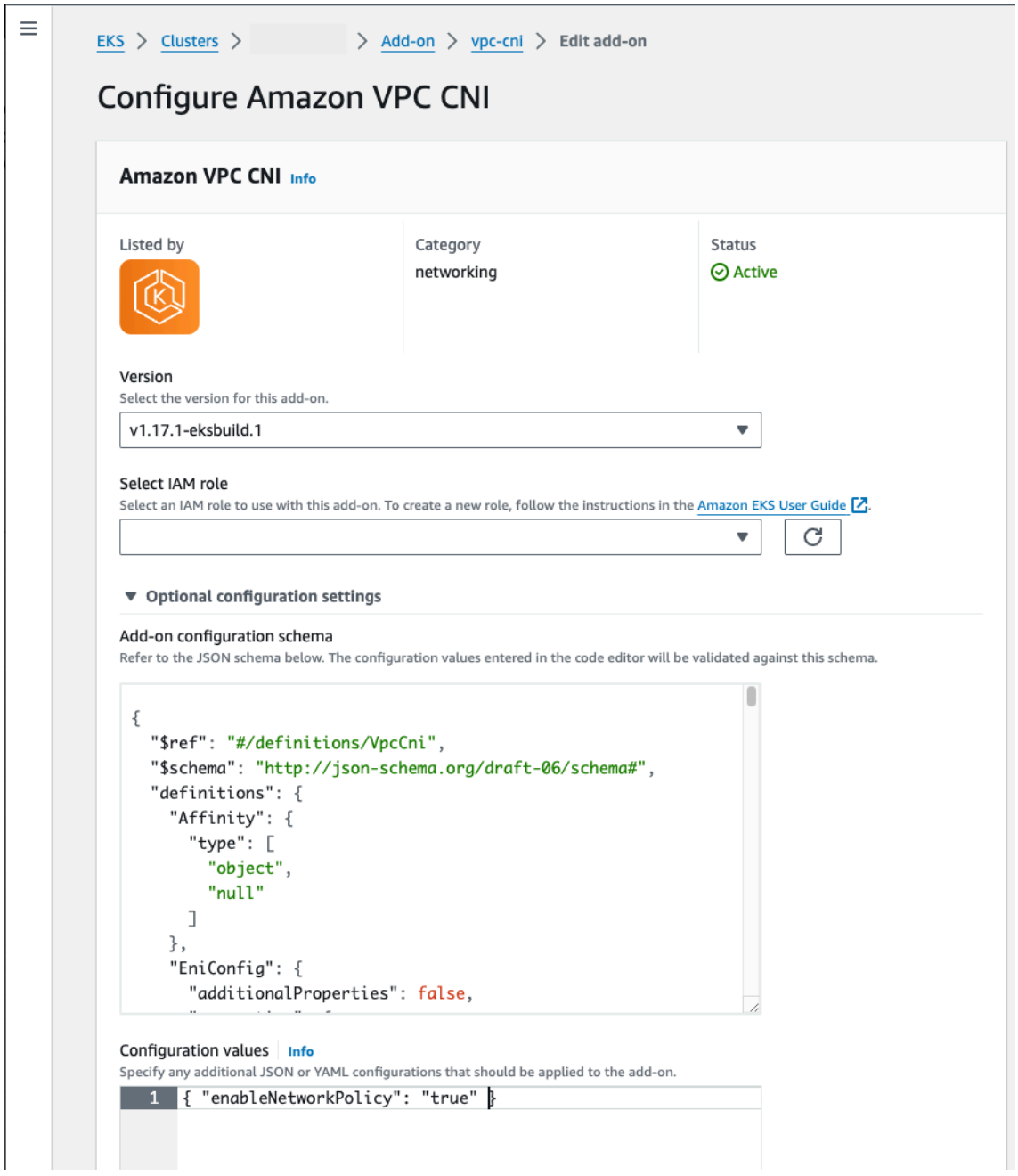
ii. Expanda Definições de configuração opcionais.

iii. Insira a chave JSON `"enableNetworkPolicy":` e o valor `"true"` em Valores da configuração. O texto resultante deve ser um objeto JSON válido. Se esse par de chave e valor for o único dado na caixa de texto,

coloque-o entre colchetes {}. O exemplo apresentado a seguir mostra que a política de rede está habilitada:

```
{ "enableNetworkPolicy": "true" }
```

A captura de tela a seguir mostra um exemplo desse cenário.



AWS CLI

- Execute o seguinte comando AWS CLI. Substitua `my-cluster` pelo nome do cluster e o ARN do perfil do IAM pelo perfil que você está usando.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni
 --addon-version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values
 '{"enableNetworkPolicy": "true"}'
```

- Complemento autogerenciado

Helm

Se você instalou o Amazon VPC CNI plugin for Kubernetes por meio do `helm`, poderá atualizar a configuração para habilitar a política de rede.

- Execute o comando a seguir para habilitar a política de rede.

```
helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

a. Abra o `amazon-vpc-cni` `ConfigMap` no editor.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

b. Adicione a linha a seguir aos `data` no `ConfigMap`.

```
enable-network-policy-controller: "true"
```

Depois de adicionar a linha, o `ConfigMap` deverá ser semelhante ao exemplo a seguir.

```
apiVersion: v1
 kind: ConfigMap
 metadata:
  name: amazon-vpc-cni
  namespace: kube-system
 data:
  enable-network-policy-controller: "true"
```

c. Abra o `aws-node` `DaemonSet` no editor.

```
kubectl edit daemonset -n kube-system aws-node
```

d. Substitua `false` por `true` no argumento do comando `--enable-network-policy=false` em `args:` no contêiner do `aws-network-policy-agent` no manifesto do daemonset do `aws-node` do VPC CNI.

```
    - args:
       - --enable-network-policy=true
```

3. Confirme que os pods do `aws-node` estão sendo executados no cluster.

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

Veja um exemplo de saída abaixo.

```
aws-node-gmqp7                                          2/2     Running   1 (24h
 ago)   24h
aws-node-prnsh                                          2/2     Running   1 (24h
 ago)   24h
```

Existem dois contêineres nos pods `aws-node` nas versão `1.14` e posteriores. Nas versões anteriores e quando a política de rede está desabilitada, há apenas um único contêiner nos pods do `aws-node`.

Agora você pode implantar políticas de rede do Kubernetes no seu cluster. Para ter mais informações, consulte Políticas de rede do Kubernetes.

Desabilitar políticas de rede

1. Liste todas as políticas de rede do Kubernetes.

```
kubectl get netpol -A
```

2. Exclua cada política de rede do Kubernetes. Você deve excluir todas as políticas de rede antes de desabilitar as políticas de rede.

```
kubectl delete netpol <policy-name>
```

3. Abra o DaemonSet aws-node em seu editor.

```
kubectl edit daemonset -n kube-system aws-node
```

4. Substitua `true` por `false` no argumento do comando `--enable-network-policy=true` em `args:` no contêiner do `aws-network-policy-agent` no manifesto do daemonset do `aws-node` do VPC CNI.

```
    - args:
       - --enable-network-policy=true
```

Denonstração Stars de política de rede

A demonstração cria um serviço de front-end, de back-end e de cliente no cluster do Amazon EKS. A demonstração também cria um interface gráfica do usuário de gerenciamento que mostra os caminhos de entrada e saída disponíveis entre cada serviço. Recomendamos que você conclua a demonstração em um cluster no qual você não executa workloads de produção.

Antes de você criar políticas de rede, todos os serviços podem se comunicar bidirecionalmente. Depois de aplicar as políticas de rede, você poderá ver que o cliente só pode se comunicar com o serviço de front-end, e o back-end só aceita tráfego do front-end.

Como executar a demonstração de política Stars

1. Aplique os serviços de front-end, de back-end, de cliente e de interface do usuário de gerenciamento:

```
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/namespace.yaml
```

```
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/management-ui.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/backend.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/frontend.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/client.yaml
```

2. Veja todos os Pods no cluster.

```
kubectl get pods -A
```

Veja um exemplo de saída abaixo.

Na saída, você deve ver pods nos namespaces mostrados na saída a seguir. Os *NOMES* dos pods e o número de pods na coluna READY são diferentes dos que aparecem na saída a seguir. Não continue até ver pods com nomes semelhantes e que todos eles tenham Running na coluna STATUS.

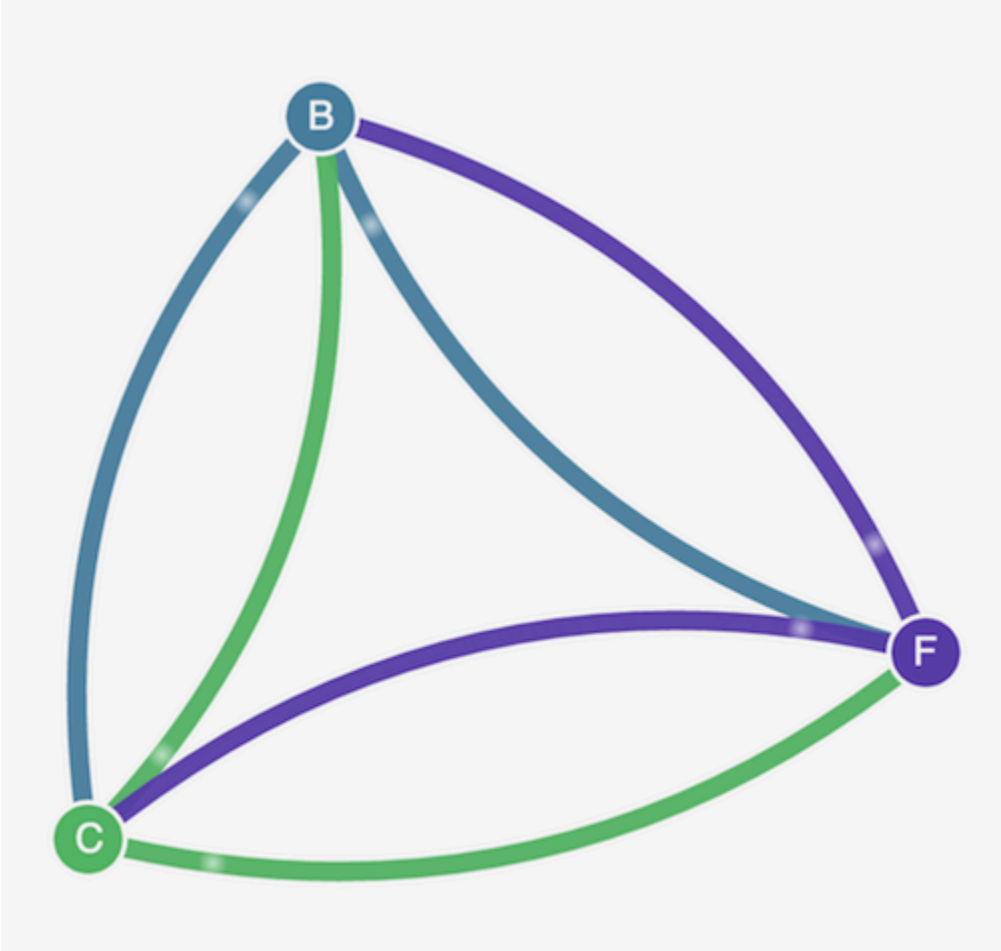
```
NAMESPACE         NAME                                       READY   STATUS
 RESTARTS   AGE
[...]
client            client-xlffc                               1/1     Running   0
       5m19s
[...]
management-ui     management-ui-qrb2g                        1/1     Running   0
       5m24s
stars             backend-sz87q                              1/1     Running   0
       5m23s
stars             frontend-cscnf                             1/1     Running   0
       5m21s
[...]
```

3. Para se conectar à interface de usuário de gerenciamento, conecte-se ao EXTERNAL-IP do serviço em execução no cluster:

```
kubectl get service/management-ui -n management-ui
```

4. Abra o navegador no local da etapa anterior. Você deve ver a interface do usuário de gerenciamento a seguir. O nó C é o serviço de cliente, o nó F é o serviço de front-end e o nó B

é o serviço de back-end. Cada nó possui acesso total à comunicação com todos os outros nós, conforme indicado pelas linhas coloridas em negrito.



5. Aplique a seguintes política de rede a ambos os namespaces `stars` e `client` para isolar os serviços um do outro:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

Você pode usar os comandos a seguir para aplicar a política a ambos os namespaces:

```
kubectl apply -n stars -f https://eksworkshop.com/beginner/120_network-policies/
calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

```
kubectl apply -n client -f https://eksworkshop.com/beginner/120_network-policies/
calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

6. Atualize o navegador. Você verá que a interface do usuário de gerenciamento não pode mais acessar nenhum dos nós, portanto, eles não são mostrados na interface do usuário.
7. Aplique as políticas de rede a seguir para permitir que a interface de usuário de gerenciamento acesse os serviços. Aplique essa política para permitir que a interface do usuário:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

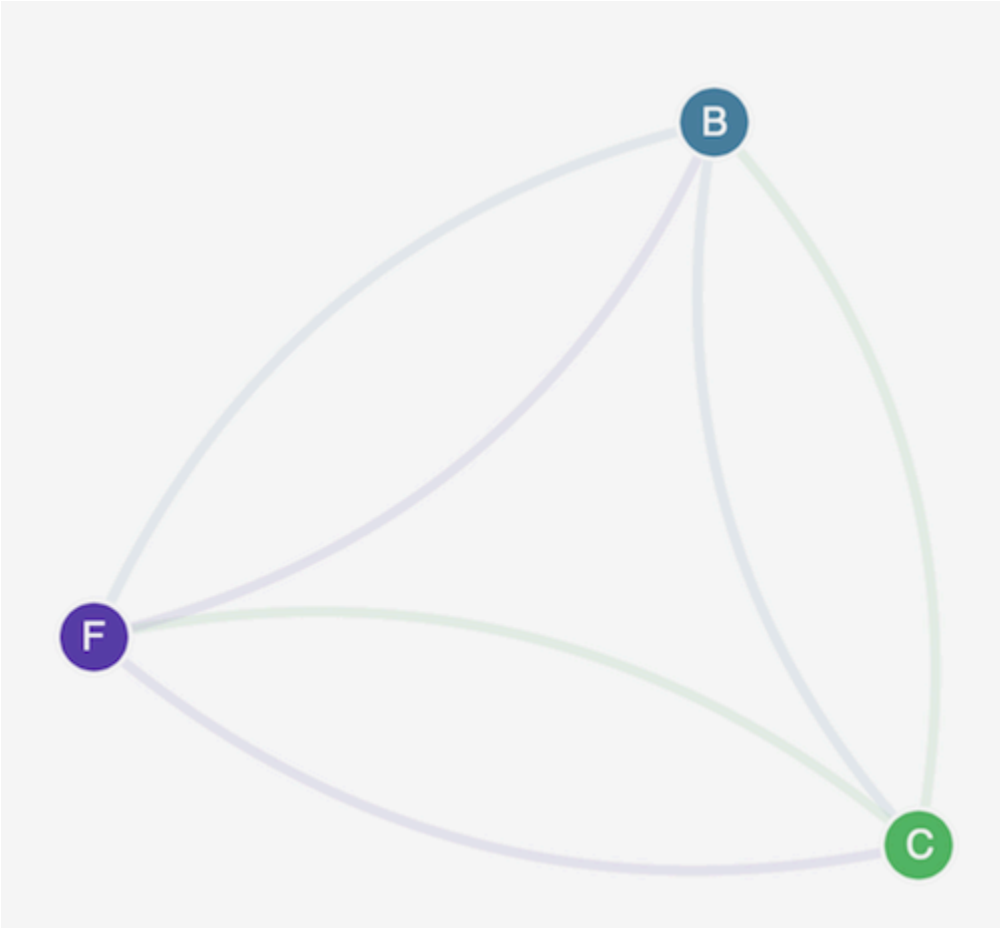
Aplique essa política para permitir que o cliente:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

Você pode usar os seguintes comandos para aplicar ambas as políticas:

```
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/apply_network_policies.files/allow-ui.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/apply_network_policies.files/allow-ui-client.yaml
```

8. Atualize o navegador. Você verá que a interface do usuário de gerenciamento pode acessar os nós novamente, mas os nós não podem se comunicar uns com os outros.

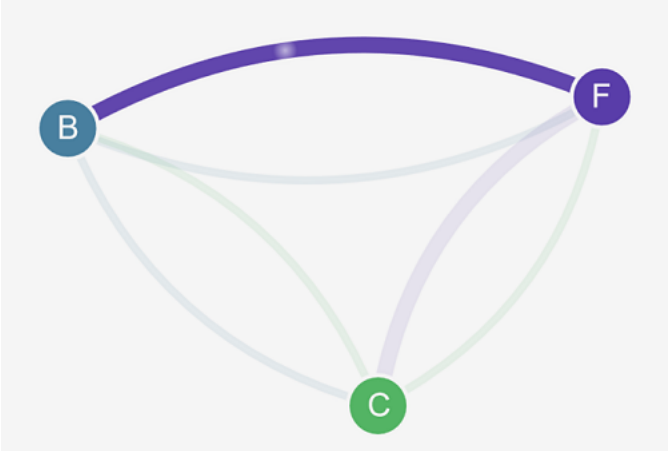


9. Aplique a seguinte política de rede para permitir o tráfego do serviço de front-end para o serviço de back-end:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: backend-policy
spec:
  podSelector:
    matchLabels:
```

```
      role: backend
  ingress:
    - from:
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
```

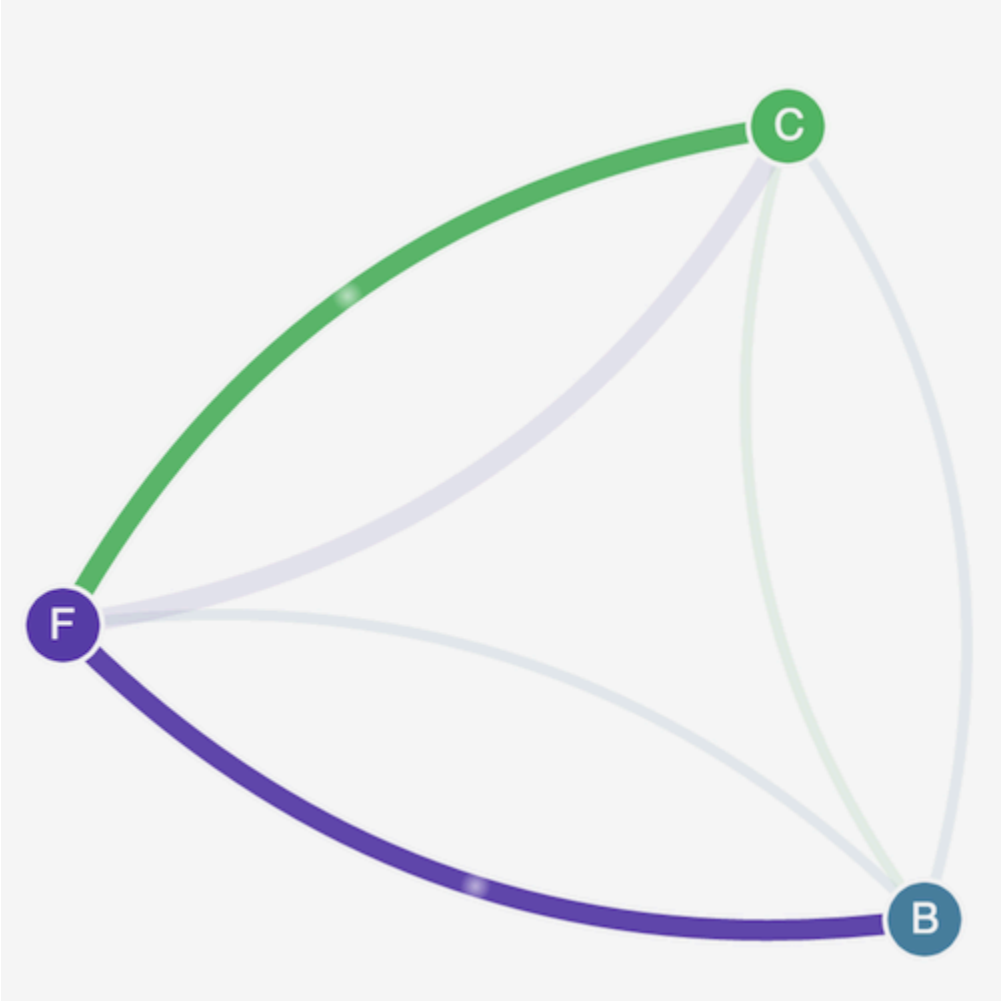
10. Atualize o navegador. Você vê que o front-end pode se comunicar com o back-end.



11. Aplique a política de rede a seguir para permitir tráfego do cliente para o serviço de front-end:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: client
      ports:
        - protocol: TCP
          port: 80
```

12. Atualize o navegador. Você vê que o cliente pode se comunicar com o serviço de front-end. O serviço de front-end ainda pode se comunicar com o serviço de back-end.



13. (Opcional) Ao concluir a demonstração, você poderá excluir os recursos dela.

```
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/frontend.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/backend.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/management-ui.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/namespace.yaml
```

Mesmo após a exclusão dos recursos, ainda pode haver endpoints de política de rede nos nós que podem interferir de maneiras inesperados nas conexões de rede do cluster. A única maneira garantida de remover essas regras é recarregar os nós ou encerrar todos os nós e reciclá-los. Para encerrar todos os nós, defina a contagem desejada do grupo de Auto Scaling como 0 e faça backup para o número desejado ou simplesmente encerre os nós.

Solucionar problemas de políticas de rede

Você pode solucionar problemas e investigar as conexões de rede que usam as políticas de rede lendo o Logs da política de rede e executando as ferramentas do eBPF SDK.

Logs da política de rede

Sejam as conexões permitidas ou negadas pelas políticas de uma rede, isso é registrado nos logs de fluxo. Os logs da política de rede de cada nó incluem os logs de fluxo para todo pod que tem uma política de rede. Os logs da política de rede são armazenados em `/var/log/aws-routed-eni/network-policy-agent.log`. O seguinte exemplo é de um arquivo de `network-policy-agent.log`:

```
{"level":"info","timestamp":"2023-05-30T16:05:32.573Z","logger":"ebpf-
client","msg":"Flow Info: ","Src
IP":"192.168.87.155","Src Port":38971,"Dest IP":"64.6.160","Dest
Port":53,"Proto":"UDP","Verdict":"ACCEPT"}
```

Os logs de política de rede estão desabilitados por padrão. Para habilitar os logs de política de rede, siga estas etapas:

> **Note**
>
> Os logs de política de rede requerem uma vCPU adicional para o contêiner `aws-network-policy-agent` no manifesto `aws-node` do daemonset do plug-in CNI da VPC.

Complemento do Amazon EKS

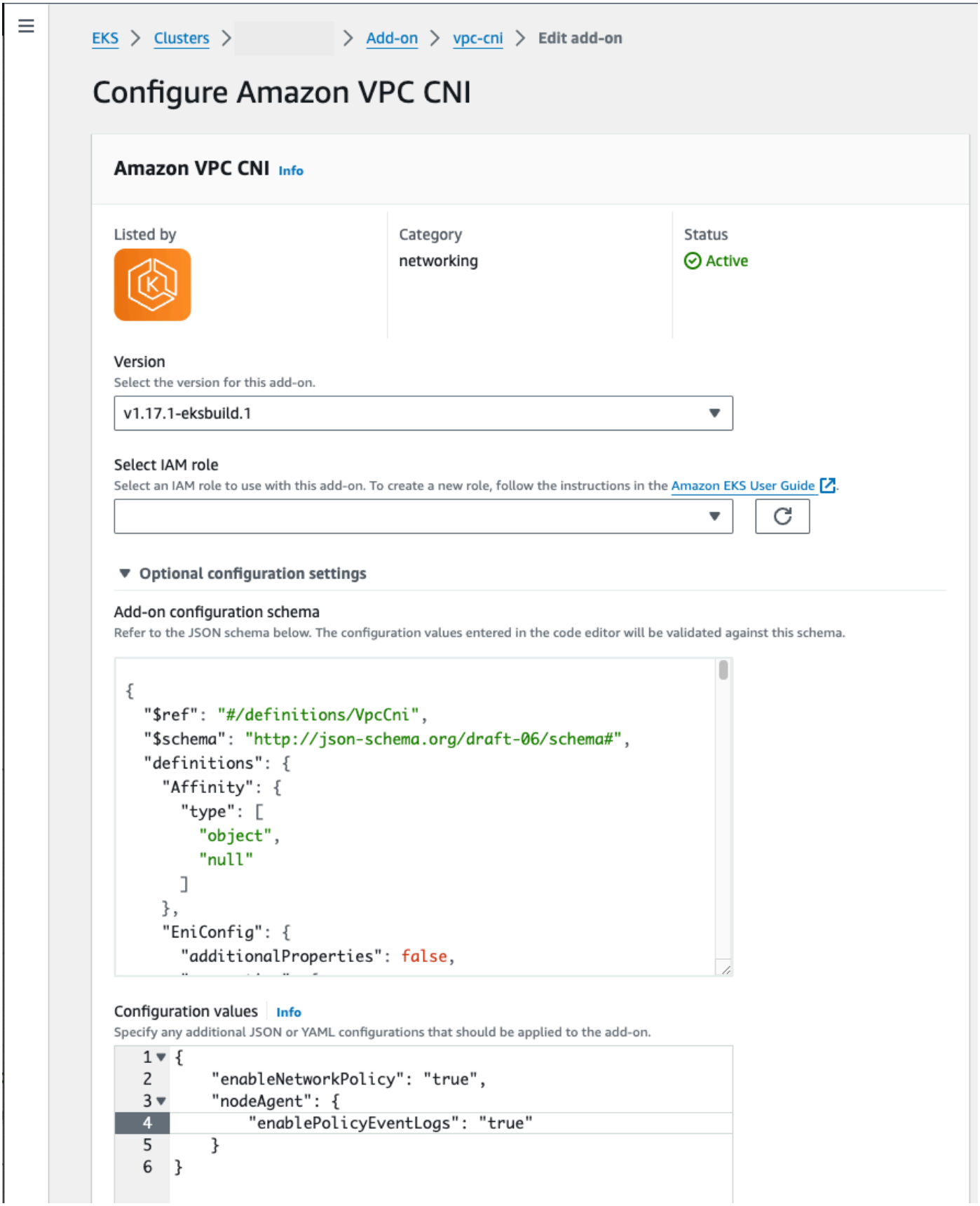
AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

2. No painel de navegação à esquerda, selecione Clusters e o nome do cluster para o qual você deseja configurar o complemento Amazon VPC CNI.
3. Escolha a guia Add-ons (Complementos).
4. Selecione a caixa no canto superior direito da caixa do complemento e depois escolha Edit (Editar).
5. Na página Configure ***name of addon*** (Configurar nome do complemento):
    a. Selecione a versão `v1.14.0-eksbuild.3` ou posterior na lista suspensa Versão.
    b. Expanda Definições de configuração opcionais.
    c. Insira a chave JSON de mais alto nível `"nodeAgent":` e o valor é um objeto com uma chave `"enablePolicyEventLogs":` e valor de `"true"` em Valores da configuração. O texto resultante deve ser um objeto JSON válido. O exemplo apresentado a seguir mostra que a política de rede e os logs de política de rede estão habilitados e que os logs de política de rede são enviados para o CloudWatch Logs:

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true"
    }
}
```

A captura de tela a seguir mostra um exemplo desse cenário.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true"
5 }
6 }

AWS CLI

- Execute o seguinte comando AWS CLI. Substitua `my-cluster` pelo nome do cluster e o ARN do perfil do IAM pelo perfil que você está usando.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true"}}'
```

Complemento autogerenciado

Helm

Caso tenha instalado o Amazon VPC CNI plugin for Kubernetes por meio do `helm`, você poderá atualizar a configuração para realizar a gravação dos logs de política de rede.

- Execute o comando a seguir para habilitar a política de rede.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

Caso tenha instalado o Amazon VPC CNI plugin for Kubernetes por meio do `kubectl`, você poderá atualizar a configuração para realizar a gravação dos logs de política de rede.

1. Abra o `aws-node DaemonSet` no editor.

```
kubectl edit daemonset -n kube-system aws-node
```

2. Substitua `false` por `true` no argumento do comando `--enable-policy-event-logs=false` em `args:` no contêiner do `aws-network-policy-agent` no manifesto do daemonset do `aws-node` do VPC CNI.

```
     - args:
```

```
        - --enable-policy-event-logs=true
```

Enviar logs de política de rede para o Amazon CloudWatch Logs

Você pode monitorar os logs da política de rede usando serviços como o Amazon CloudWatch Logs. Você pode usar os métodos a seguir para enviar os logs da política de rede para o CloudWatch Logs.

Para clusters do EKS, os logs da política estarão localizados em `/aws/eks/cluster-name/cluster/` e para clusters do K8S autogerenciados, os logs serão colocados em `/aws/k8s-cluster/cluster/`.

Enviar logs da política de rede com o Amazon VPC CNI plugin for Kubernetes

Se você habilitar uma política de rede, um segundo contêiner será adicionado aos pods do `aws-node` para um agente do nó. Esse agente do nó pode enviar os logs da política de rede para o CloudWatch Logs.

> **Note**
>
> Somente os logs da política de rede são enviados pelo agente do nó. Outros logs feitos pelo VPC CNI não são incluídos.

Pré-requisitos

- Adicione as permissões a seguir como uma seção ou uma política separada ao perfil do IAM que você está usando para o VPC CNI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Complemento do Amazon EKS

AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, selecione Clusters e o nome do cluster para o qual você deseja configurar o complemento Amazon VPC CNI.
3. Escolha a guia Add-ons (Complementos).
4. Selecione a caixa no canto superior direito da caixa do complemento e depois escolha Edit (Editar).
5. Na página Configure ***name of addon*** (Configurar nome do complemento):

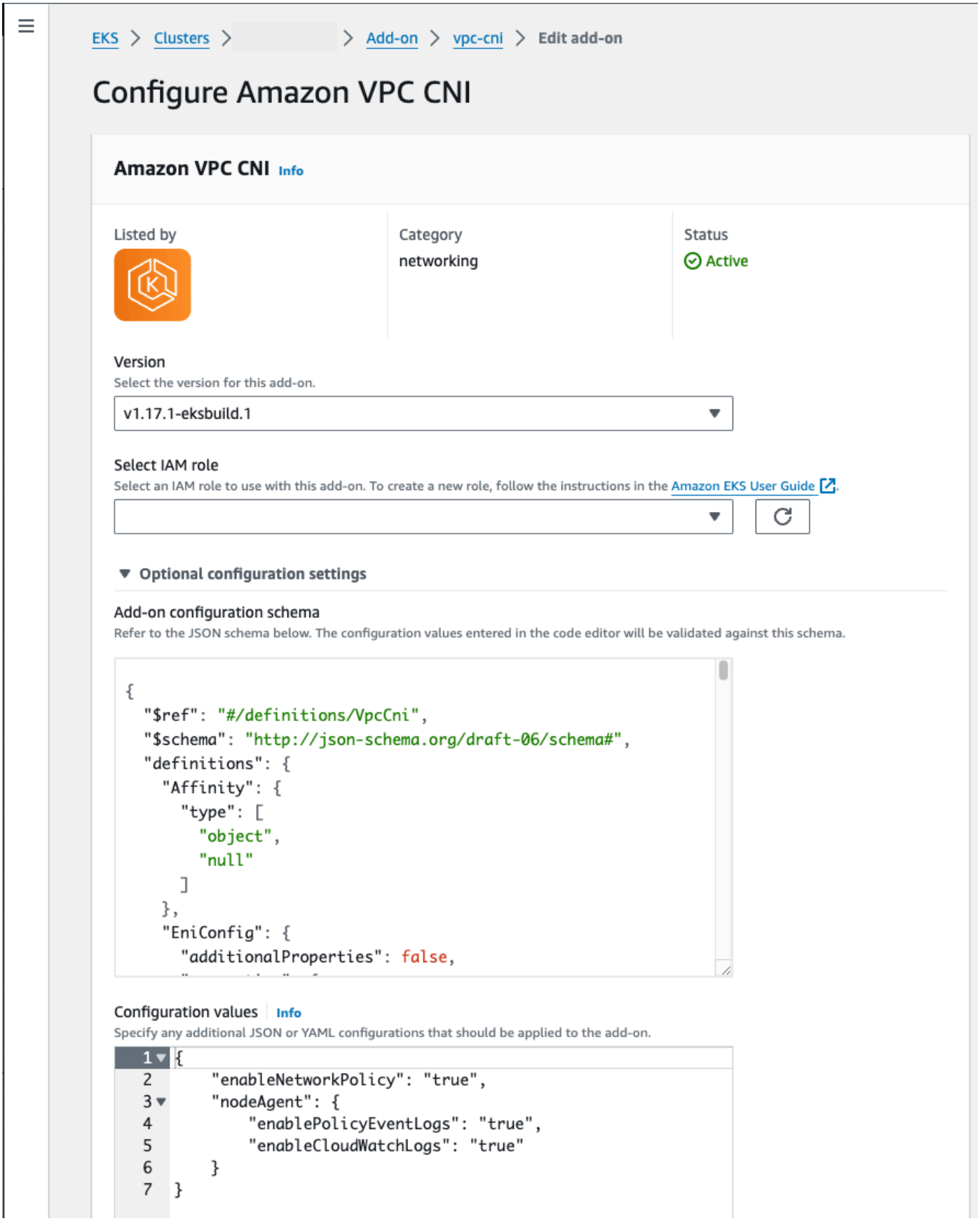
   a. Selecione a versão `v1.14.0-eksbuild.3` ou posterior na lista suspensa Versão.

   b. Expanda Definições de configuração opcionais.

   c. Insira a chave JSON de mais alto nível `"nodeAgent":` e o valor é um objeto com uma chave `"enableCloudWatchLogs":` e valor de `"true"` em Valores da configuração. O texto resultante deve ser um objeto JSON válido. O exemplo apresentado a seguir mostra que a política de rede e os logs de política de rede estão habilitados e que os logs são enviados para o CloudWatch Logs:

   ```
   {
       "enableNetworkPolicy": "true",
       "nodeAgent": {
           "enablePolicyEventLogs": "true",
           "enableCloudWatchLogs": "true",
       }
   }
   ```

A captura de tela a seguir mostra um exemplo desse cenário.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true",
5 "enableCloudWatchLogs": "true"
6 }
7 }

AWS CLI

- Execute o seguinte comando AWS CLI. Substitua `my-cluster` pelo nome do cluster e o ARN do perfil do IAM pelo perfil que você está usando.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true", "enableCloudWatchLogs": "true"}}'
```

Complemento autogerenciado

Helm

Se você instalou o Amazon VPC CNI plugin for Kubernetes por meio do `helm`, poderá atualizar a configuração para enviar os logs de política de rede para o CloudWatch Logs.

- Execute o comando apresentado a seguir para habilitar logs de política de rede e enviá-los ao CloudWatch Logs.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set
 nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. Abra o `aws-node DaemonSet` no editor.

```
kubectl edit daemonset -n kube-system aws-node
```

2. Substitua `false` por `true` nos dois argumentos do comando `--enable-policy-event-logs=false` e `--enable-cloudwatch-logs=false` em `args:` no contêiner `aws-network-policy-agent` no manifesto `aws-node` do daemonset do plug-in CNI da VPC.

```
    - args:
       - --enable-policy-event-logs=true
```

```
        - --enable-cloudwatch-logs=true
```

Enviar logs de política de rede com o daemonset Fluent Bit

Se você estiver usando Fluent Bit em um daemonset para enviar os logs dos nós, poderá adicionar configurações para incluir os logs das políticas de rede. Você pode usar o seguinte exemplo de configuração:

```
    [INPUT]
        Name              tail
        Tag               eksnp.*
        Path              /var/log/aws-routed-eni/network-policy-agent*.log
        Parser            json
        DB                /var/log/aws-routed-eni/flb_npagent.db
        Mem_Buf_Limit     5MB
        Skip_Long_Lines   On
        Refresh_Interval  10
```

SDK do eBPF incluído

O SDK do Amazon VPC CNI plugin for Kubernetes instala o conjunto de ferramentas do SDK do eBPF nos nós. Você pode usar as ferramentas do SDK do eBPF para identificar problemas de políticas de rede. Por exemplo, o comando a seguir lista os programas que estão sendo executados no nó.

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

Para executar esse comando, você pode usar qualquer método de conexão com o nó.

Políticas de rede do Kubernetes

Para implementar as políticas de rede do Kubernetes, você cria objetos de Kubernetes `NetworkPolicy` e implanta-os no cluster. O escopo dos objetos de `NetworkPolicy` é definido como um namespace. Você implementa políticas para permitir ou negar tráfego entre os Pods com base em seletores de rótulos, namespaces e intervalos de endereços IP. Para obter mais informações sobre a criação de objetos de `NetworkPolicy`, consulte Políticas de rede na documentação do Kubernetes.

A aplicação dos objetos de `NetworkPolicy` do Kubernetes é implementada usando o Extended Berkeley Packet Filter (eBPF). Em relação a implementações baseadas em `iptables`, ele oferece

menor latência e características de performance, incluindo menor utilização da CPU e prevenção de consultas sequenciais. Além disso, as sondas do eBPF fornecem acesso a dados contextuais ricos que ajudam a depurar problemas complexos no nível do kernel e a melhorar a observabilidade. O Amazon EKS é compatível com um exportador baseado no eBPF que aproveita as sondas para registrar em log os resultados da política em cada nó e exportar os dados para coletores de log externos para auxiliar na depuração. Para obter mais informações, consulte a documentação do eBPF.

Rede personalizada para pods

Por padrão, quando o Amazon VPC CNI plugin for Kubernetes cria interfaces de rede elásticas secundárias (interfaces de rede) para o nó do Amazon EC2, ele as cria na mesma sub-rede que a interface de rede primária do nó. Ele também associa os mesmos grupos de segurança à interface de rede secundária que estão associados à interface de rede primária. Por uma ou mais das razões a seguir, você pode querer que o plugin crie interfaces de rede secundárias em uma sub-rede diferente ou associar grupos de segurança diferentes às interfaces de rede secundárias, ou ambos:

- Existe um número limitado de endereços `IPv4` disponíveis na sub-rede na qual a interface de rede primária se encontra. Isso pode limitar o número de Pods que podem ser criados na sub-rede. Usando uma sub-rede diferente para interfaces de rede secundárias, é possível aumentar o número de endereços `IPv4` disponíveis para Pods.
- Por motivos de segurança, os Pods talvez precisem usar diferentes grupos de segurança ou sub-redes do que os da interface de rede primária do nó.
- Os nós são configurados em sub-redes públicas, e você deseja colocar os Pods em sub-redes privadas. A tabela de rotas que está associada a uma sub-rede pública inclui uma rota para um gateway da Internet. A tabela de rotas que está associada a uma sub-rede privada não inclui uma rota para um gateway da Internet.

Considerações

- Com redes personalizadas habilitadas, nenhum endereço IP atribuído à interface de rede primária é atribuído a Pods. Apenas endereços IP de interfaces de rede secundárias são atribuídos a `Pods`.
- Se o seu cluster utilizar a família `IPv6`, não será possível utilizar redes personalizadas.
- Se você planeja utilizar redes personalizadas somente para aliviar a exaustão de endereços `IPv4`, pode criar um cluster utilizando a família `IPv6`. Para ter mais informações, consulte Endereços `IPv6` para clusters, Pods e services.

- Mesmo que Pods implantados em sub-redes especificadas para interfaces de rede secundárias possam utilizar sub-redes e grupos de segurança diferentes da interface de rede primária do nó, as sub-redes e os grupos de segurança devem estar na mesma VPC do nó.

Pré-requisitos

- Familiaridade com o procedimento do Amazon VPC CNI plugin for Kubernetes de criar interfaces de rede secundárias e atribuir endereços IP aos Pods. Para saber mais, consulte ENI Allocation (Alocação de ENIs) no GitHub.
- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.
- Convém concluir as etapas neste tópico em um shell Bash. Se não estiver utilizando um shell Bash, alguns comandos de script, como caracteres de continuação de linha e a forma como as variáveis são definidas e utilizadas, exigirão o ajuste do seu shell. Além disso, as regras de citação e de escape do seu shell podem ser diferentes. Para obter mais informações, consulte Uso de aspas com strings na AWS CLI no Guia do usuário da AWS Command Line Interface.

Para este tutorial, convém utilizar os *`example values`*, exceto onde indicado para substituí-los. É possível substituir qualquer *`example value`* ao realizar as etapas para um cluster em produção. Convém que todas as etapas sejam concluídas no mesmo terminal. Isso porque variáveis são definidas e utilizadas ao longo das etapas e não existirão em terminais diferentes.

Os comandos neste tópico são formatados usando as convenções listadas em Como usar os exemplos da AWS CLI. Se você estiver executando comandos usando a linha de comando em

recursos que se encontram em uma Região da AWS que não é a Região da AWS padrão definida no perfil da AWS CLI que você está usando, será necessário adicionar **`--region region-code`** aos comandos.

Quando quiser implantar redes personalizadas no seu cluster de produção, pule para Etapa 2: Configurar sua VPC.

Etapa 1: Criar uma VPC de teste e um cluster

Para criar um cluster

Os procedimentos a seguir ajudam a criar uma VPC de teste e um cluster e a configurar redes personalizadas para esse cluster. Não recomendamos utilizar o cluster de teste para workloads de produção, pois vários recursos não relacionados que você pode utilizar no seu cluster de produção não são abordados neste tópico. Para ter mais informações, consulte Criar um cluster do Amazon EKS..

1. Defina algumas variáveis para uso nas etapas restantes.

```
export cluster_name=my-custom-networking-cluster
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. Crie uma VPC.

   1. Crie uma VPC utilizando um modelo AWS CloudFormation do Amazon EKS.

```
aws cloudformation create-stack --stack-name my-eks-custom-networking-vpc \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --parameters ParameterKey=VpcBlock,ParameterValue=192.168.0.0/24 \
  ParameterKey=PrivateSubnet01Block,ParameterValue=192.168.0.64/27 \
  ParameterKey=PrivateSubnet02Block,ParameterValue=192.168.0.96/27 \
  ParameterKey=PublicSubnet01Block,ParameterValue=192.168.0.0/27 \
  ParameterKey=PublicSubnet02Block,ParameterValue=192.168.0.32/27
```

      A pilha AWS CloudFormation leva alguns minutos para ser criada. Para verificar o status de implantação da pilha, execute o comando a seguir.

```
aws cloudformation describe-stacks --stack-name my-eks-custom-networking-vpc --
query Stacks\[\].StackStatus  --output text
```

Não prossiga para a próxima etapa até que a saída do comando seja `CREATE_COMPLETE`.

2. Defina variáveis com os valores dos IDs das sub-redes privadas criadas pelo modelo.

```
subnet_id_1=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet02'].PhysicalResourceId" --output text)
```

3. Defina variáveis com as zonas de disponibilidade das sub-redes que foram recuperadas na etapa anterior.

```
az_1=$(aws ec2 describe-subnets --subnet-ids $subnet_id_1 --query
 'Subnets[*].AvailabilityZone' --output text)
az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --query
 'Subnets[*].AvailabilityZone' --output text)
```

3. Crie um perfil do IAM de cluster.

a. Execute o seguinte comando para criar um arquivo JSON de política de confiança do IAM:

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. Crie o perfil do IAM do cluster do Amazon EKS. Se necessário, prefixe `eks-cluster-role-trust-policy.json` com o caminho no computador no qual você gravou o arquivo

na etapa anterior. O comando associa a política de confiança criada na etapa anterior à função. Para criar um perfil do IAM, a entidade principal do IAM que estiver criando o perfil deverá ser atribuída à seguinte ação `iam:CreateRole` (permissão):

```
aws iam create-role --role-name myCustomNetworkingAmazonEKSClusterRole --
assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

c. Anexe a política gerenciada do Amazon EKS denominada `AmazonEKSClusterPolicy` à função. Para anexar uma política do IAM a uma entidade principal do IAM a entidade principal do IAM que está anexando a política deve receber uma das seguintes ações do IAM (permissões): `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name myCustomNetworkingAmazonEKSClusterRole
```

4. Crie um cluster do Amazon EKS e configure seu dispositivo para se comunicar com ele.

a. Crie um cluster.

```
aws eks create-cluster --name my-custom-networking-cluster \
   --role-arn arn:aws:iam::$account_id:role/
myCustomNetworkingAmazonEKSClusterRole \
   --resources-vpc-config subnetIds=$subnet_id_1","$subnet_id_2
```

> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para ter mais informações, consulte Insufficient capacity (Capacidade insuficiente).

b. A criação do cluster demora alguns minutos. Para verificar o status de implantação do cluster, execute o comando a seguir.

```
aws eks describe-cluster --name my-custom-networking-cluster --query
 cluster.status
```

Não prossiga para a próxima etapa até que a saída do comando seja `"ACTIVE"`.

c. Configure `kubectl` para se comunicar com o cluster.

```
aws eks update-kubeconfig --name my-custom-networking-cluster
```

Etapa 2: Configurar sua VPC

Este tutorial requer a VPC que foi criada em Etapa 1: Criar uma VPC de teste e um cluster. Para um cluster de produção, ajuste as etapas de acordo com a VPC, substituindo todos os *example values* por seu próprio.

1. Confirme se a sua versão instalada do Amazon VPC CNI plugin for Kubernetes é a mais recente. Para determinar a versão mais recente do tipo de complemento do Amazon EKS e atualizar sua versão para ela, consulte Atualizar um complemento do Amazon EKS. Para determinar a versão mais recente do tipo de complemento autogerenciado e atualizar sua versão para ela, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.
2. Recupere o ID da VPC do seu cluster e armazene-o em uma variável para uso em etapas posteriores. Para um cluster de produção, substitua *my-custom-networking-cluster* pelo nome do seu cluster.

```
vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query
 "cluster.resourcesVpcConfig.vpcId" --output text)
```

3. Associe um bloco de Encaminhamento Entre Domínios Sem Classificação (CIDR) adicional à VPC do seu cluster. O bloco CIDR não pode se sobrepor a nenhum bloco CIDR existente associado.

   1. Visualize os blocos CIDR atuais que estão associados à sua VPC.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id \
    --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

Veja um exemplo de saída abaixo.

```
---------------------------------
|          DescribeVpcs         |
+-----------------+-------------+
|    CIDRBlock    |    State    |
+-----------------+-------------+
|  192.168.0.0/24 |  associated |
+-----------------+-------------+
```

2. Associe um bloco CIDR adicionais à sua VPC. Para obter mais informações, consulte Associar blocos CIDR IPv4 adicionais à sua VPC, no Guia do usuário da Amazon VPC.

```
aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 192.168.1.0/24
```

3. Confirme se o novo bloco está associado.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --query
 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

Veja um exemplo de saída abaixo.

```
---------------------------------
|          DescribeVpcs         |
+-----------------+-------------+
|    CIDRBlock    |    State    |
+-----------------+-------------+
|  192.168.0.0/24 |  associated |
|  192.168.1.0/24 |  associated |
+-----------------+-------------+
```

Não continue na próxima etapa até que o `State` do novo bloco CIDR seja `associated`.

4. Crie quantas sub-redes quiser utilizar em cada zona de disponibilidade na qual suas sub-redes existentes estejam. Especifique um bloco CIDR que esteja dentro do bloco CIDR que você associou à sua VPC em uma etapa anterior.

1. Crie novas sub-redes. As sub-redes devem ser criadas em um bloco CIDR de VPC diferente daquele em que as sub-redes existentes se encontram, mas nas mesmas zonas de

disponibilidade dessas sub-redes existentes. Neste exemplo, uma sub-rede é criada no novo bloco CIDR em cada zona de disponibilidade na qual as sub-redes privadas atuais existem. Os IDs das sub-redes criadas são armazenados em variáveis para uso em etapas posteriores. Os valores de `Name` correspondem aos valores atribuídos às sub-redes criadas utilizando o modelo VPC do Amazon EKS em uma etapa anterior. Não são necessários nomes. É possível utilizar nomes diferentes.

```
new_subnet_id_1=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone
 $az_1 --cidr-block 192.168.1.0/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-
custom-networking-vpc-PrivateSubnet01},{Key=kubernetes.io/role/internal-
elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
new_subnet_id_2=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone
 $az_2 --cidr-block 192.168.1.32/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-
custom-networking-vpc-PrivateSubnet02},{Key=kubernetes.io/role/internal-
elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
```

> **Important**
>
> Por padrão, suas novas sub-redes estão implicitamente associadas a tabela de rotas principal da sua VPC. Essa tabela de rotas permite a comunicação entre todos os recursos que estão implantados na VPC. Porém, ela não permite comunicação com recursos que têm endereços IP fora dos blocos CIDR associados à sua VPC. É possível associar sua própria tabela de rotas a sub-redes para alterar esse comportamento. Para obter mais informações, consulte Tabelas de rotas de sub-rede, no Guia do usuário da Amazon VPC.

2. Visualize as sub-redes atuais na sua VPC.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" \
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

Veja um exemplo de saída abaixo.

```
-------------------------------------------------------------------
|                          DescribeSubnets                        |
+------------------+--------------------+---------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId          |
+------------------+--------------------+---------------------------+
|  us-west-2d      |  192.168.0.0/27    |     subnet-example1       |
|  us-west-2a      |  192.168.0.32/27   |     subnet-example2       |
|  us-west-2a      |  192.168.0.64/27   |     subnet-example3       |
|  us-west-2d      |  192.168.0.96/27   |     subnet-example4       |
|  us-west-2a      |  192.168.1.0/27    |     subnet-example5       |
|  us-west-2d      |  192.168.1.32/27   |     subnet-example6       |
+------------------+--------------------+---------------------------+
```

É possível ver que as sub-redes no bloco CIDR `192.168.1.0` que você criou estão nas mesmas zonas de disponibilidade que as sub-redes do bloco CIDR `192.168.0.0`.

Etapa 3: Configuração de recursos do Kubernetes

Para configurar o recursos do Kubernetes

1. Defina a variável de ambiente `AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG` como `true` no `aws-node DaemonSet`.

   ```
   kubectl set env daemonset aws-node -n kube-system
    AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
   ```

2. Recupere o ID do grupo de segurança do cluster e armazene-o em uma variável para usar na próxima etapa. O Amazon EKS criará automaticamente esse grupo de segurança quando você criar seu cluster.

   ```
   cluster_security_group_id=$(aws eks describe-cluster --name $cluster_name --query
    cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
   ```

3. Crie um recurso `ENIConfig` personalizado para cada sub-rede na qual você deseja programar Pods.

   a. Crie um arquivo exclusivo para cada configuração de interface de rede elástica.

Os comandos a seguir criam arquivos `ENIConfig` separados para as duas sub-redes criadas em uma etapa anterior. O valor de `name` deve ser exclusivo. O nome é o mesmo da zona de disponibilidade na qual a sub-rede está localizada. O grupo de segurança do cluster é atribuído a `ENIConfig`.

```
cat >$az_1.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
EOF
```

```
cat >$az_2.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_2
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_2
EOF
```

Para um cluster de produção, é possível fazer as seguintes alterações nos comandos anteriores:

- Substitua *`$cluster_security_group_id`* pelo ID de um grupo de segurança existente que você deseja utilizar para cada `ENIConfig`.
- Sempre que possível, convém definir seu `ENIConfigs` com o mesmo nome que o da zona de disponibilidade para a qual você usará o `ENIConfig`. Talvez você precise utilizar nomes para os `ENIConfigs` diferentes dos nomes das zonas de disponibilidade por vários motivos. Por exemplo, se você tiver mais de duas sub-redes na mesma zona de disponibilidade e quiser utilizá-las com redes personalizadas, precisará de vários `ENIConfigs` para a mesma zona de disponibilidade. Como cada `ENIConfig` requer um

nome exclusivo, não é possível nomear mais do que um dos seus `ENIConfigs` usando o nome da zona de disponibilidade.

Se os nomes de `ENIConfig` não forem todos iguais aos nomes de zonas de disponibilidade, substitua *`$az_1`* e *`$az_2`* por seus próprios nomes nos comandos anteriores e anote esses nós com o `ENIConfig` mais adiante neste tutorial.

> **Note**
>
> Se um grupo de segurança válido não for especificado para uso com um cluster de produção e você estiver usando:
>
> - a versão `1.8.0` ou posterior do Amazon VPC CNI plugin for Kubernetes, os grupos de segurança associados à interface de rede elástica primária do nó serão utilizados.
> - uma versão do Amazon VPC CNI plugin for Kubernetes anterior a `1.8.0`, o grupo de segurança padrão para a VPC será atribuído a interfaces de rede secundárias.

> **Important**
>
> - `AWS_VPC_K8S_CNI_EXTERNALSNAT=false` é uma configuração padrão na configuração do plug-in Amazon VPC CNI para Kubernetes. Se você estiver utilizando essa configuração padrão, o tráfego destinado a endereços IP que estiverem fora de um dos blocos CIDR associados à sua VPC utilizará os grupos de segurança e as sub-redes da interface de rede primária do nó. As sub-redes e os grupos de segurança definidos em `ENIConfigs` que forem utilizados para criar interfaces de rede secundárias não serão utilizados para esse tráfego. Para obter mais informações sobre essa configuração, consulte SNAT para Pods.
> - Se você também utilizar grupos de segurança para Pods, o grupo de segurança especificado em uma `SecurityGroupPolicy` será utilizado no lugar do grupo de segurança especificado em `ENIConfigs`. Para ter mais informações, consulte Grupos de segurança do Pods.

b. Aplique cada arquivo de recursos personalizados que você criou ao seu cluster com os seguintes comandos:

```
kubectl apply -f $az_1.yaml
kubectl apply -f $az_2.yaml
```

4. Confirme se as `ENIConfigs` foram criadas.

```
kubectl get ENIConfigs
```

Veja um exemplo de saída abaixo.

```
NAME         AGE
us-west-2a   117s
us-west-2d   105s
```

5. Se estiver habilitando redes personalizadas em um cluster de produção e tiver especificado para as suas `ENIConfigs` nomes diferentes dos da zona de disponibilidade para a qual você as estiver utilizando, pule para a próxima etapa para implantar nós do Amazon EC2.

Habilite o Kubernetes para aplicar automaticamente `ENIConfig` para uma zona de disponibilidade a todos os novos nós do Amazon EC2 que forem criados no cluster.

1. No caso do cluster de teste neste tutorial, pule para a próxima etapa.

Para um cluster de produção, verifique se uma annotation com a chave `k8s.amazonaws.com/eniConfig` para a variável de ambiente `ENI_CONFIG_ANNOTATION_DEF` existe na especificação de contêiner do `aws-node` `DaemonSet`.

```
kubectl describe daemonset aws-node -n kube-system | grep
 ENI_CONFIG_ANNOTATION_DEF
```

Se a saída for retornada, significa que a anotação existe. Se nenhuma saída for retornada, a variável não está definida. Para um cluster de produção, é possível utilizar essa configuração ou a configuração da etapa a seguir. Se você utilizar essa configuração, ela substituirá a configuração da etapa a seguir. Neste tutorial, é utilizada a configuração da próxima etapa.

2. Atualize seu `aws-node` `DaemonSet` para aplicar automaticamente a `ENIConfig` de uma zona de disponibilidade a todos os novos nós do Amazon EC2 que forem criados no seu cluster.

```
kubectl set env daemonset aws-node -n kube-system
 ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
```

Etapa 4: Implantar nós do Amazon EC2

Para implantar nós do Amazon EC2

1. Crie uma função do IAM para o nó.

   a. Execute o seguinte comando para criar um arquivo JSON de política de confiança do IAM:

   ```
   cat >node-role-trust-relationship.json <<EOF
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "ec2.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   EOF
   ```

   b. Execute o seguinte comando para definir uma variável para o nome da sua função. Você pode substituir *myCustomNetworkingAmazonEKSNodeRole* por qualquer nome que escolher.

   ```
   export node_role_name=myCustomNetworkingAmazonEKSNodeRole
   ```

   c. Crie o perfil do IAM e armazene o nome do recurso da Amazon (ARN) retornado em uma variável para uso em uma etapa posterior.

   ```
   node_role_arn=$(aws iam create-role --role-name $node_role_name --assume-role-
   policy-document file://"node-role-trust-relationship.json" \
       --query Role.Arn --output text)
   ```

   d. Anexe três políticas do IAM gerenciadas necessárias ao perfil do IAM.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name $node_role_name
```

> **Important**
>
> Para simplificar este tutorial, a política `AmazonEKS_CNI_Policy` é anexada ao perfil do IAM do nó. No entanto, em um cluster de produção, recomendamos que você anexe a política a um perfil do IAM separado que não seja usado com o Amazon VPC CNI plugin for Kubernetes. Para ter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).

2. Crie um dos seguintes tipos de grupos de nós. Para determinar o tipo de instância a ser implantado, consulte Escolher um tipo de instância de nó do Amazon EC2 ideal. Neste tutorial, conclua a opção Gerenciado, Sem um modelo de inicialização ou com um modelo de inicialização sem um ID de AMI especificado. Se for utilizar o grupo de nós para workloads de produção, convém familiarizar-se com todas as opções de grupos de nós gerenciados e autogerenciados antes de implantar o grupo de nós.

   - Managed (Gerenciado) – Implante seu grupo de nós usando uma das seguintes opções:
     - Sem um modelo de inicialização ou com um modelo de inicialização sem um ID de AMI especificado: execute o comando a seguir. Neste tutorial, usaremos o *`example values`*. Para um grupo de nós em produção, substitua todos os *`example values`* por seus próprios valores. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.

```
aws eks create-nodegroup --cluster-name $cluster_name --nodegroup-name my-
nodegroup \
```

```
    --subnets $subnet_id_1 $subnet_id_2 --instance-types t3.medium --node-role
$node_role_arn
```

- Com um modelo de inicialização com um ID de AMI especificado

  1. Determine o número máximo de Pods que o Amazon EKS recomenda para os nós. Siga as instruções em Máximo recomendado de Pods do Amazon EKS para cada tipo de instância do Amazon EC2, adicionando **`--cni-custom-networking-enabled`** à etapa 3 nesse tópico. Anote a saída para uso na próxima etapa.
  2. No seu modelo de inicialização, especifique um ID de AMI otimizado do Amazon EKS ou uma AMI personalizada criada a partir da AMI otimizada do Amazon EKS e, em seguida implante o grupo de nós usando um modelo de inicialização e forneça os seguintes dados do usuário no modelo de lançamento: Esses dados do usuário transmitem argumentos para o arquivo `bootstrap.sh`. Para obter mais informações sobre o arquivo de bootstrap, consulte bootstrap.sh no GitHub. Você pode substituir *20* pelo valor da etapa anterior (recomendado) ou por seu próprio valor.

     ```
     /etc/eks/bootstrap.sh my-cluster --use-max-pods false --kubelet-extra-args
      '--max-pods=20'
     ```

     Se você criou uma AMI personalizada que não foi criada a partir da AMI otimizada do Amazon EKS, precisa criar a configuração você mesmo.

- Autogerenciado

  1. Determine o número máximo de Pods que o Amazon EKS recomenda para os nós. Siga as instruções em Máximo recomendado de Pods do Amazon EKS para cada tipo de instância do Amazon EC2, adicionando **`--cni-custom-networking-enabled`** à etapa 3 nesse tópico. Anote a saída para uso na próxima etapa.
  2. Implante o grupo de nós usando as instruções em Criar nós autogerenciados do Amazon Linux. Especifique o seguinte texto para o parâmetro BootstrapArguments: Você pode substituir *20* pelo valor da etapa anterior (recomendado) ou por seu próprio valor.

     ```
     --use-max-pods false --kubelet-extra-args '--max-pods=20'
     ```

> **Note**
>
> Se quiser que os nós em um cluster de produção ofereçam suporte a um número significativamente maior de Pods, execute o script em Máximo recomendado de Pods

do Amazon EKS para cada tipo de instância do Amazon EC2 novamente. Adicione também a opção **--cni-prefix-delegation-enabled** ao comando. Por exemplo, *110* é retornado para um tipo de instância `m5.large`. Para obter instruções sobre como habilitar esse recurso, consulte Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2. É possível usar esse recurso com redes personalizadas.

A criação do grupo de nós demora vários minutos. Você pode verificar o status da criação de um grupo de nós gerenciados com o seguinte comando:

```
aws eks describe-nodegroup --cluster-name $cluster_name --nodegroup-name my-nodegroup --query nodegroup.status --output text
```

Não continue na próxima etapa até que a saída recebida esteja `ACTIVE`.

3. No tutorial, é possível ignorar essa etapa.

Para um cluster de produção, se você definir para as suas `ENIConfigs` os mesmos nomes que os da zona de disponibilidade para a qual você as estiver utilizando, deverá anotar seus nós com o nome de `ENIConfig` que deve ser utilizado com o nó. Essa etapa não será necessária se você tiver apenas uma sub-rede em cada zona de disponibilidade e tiver definido suas `ENIConfigs` com os mesmos nomes das zonas de disponibilidade. Isso ocorre porque oAmazon VPC CNI plugin for Kubernetes associa automaticamente a `ENIConfig` correta com o nó quando você o habilitou a fazer isso em uma etapa anterior.

a. Obtenha a lista de nós no seu cluster.

```
kubectl get nodes
```

Veja um exemplo de saída abaixo.

```
NAME                                           STATUS   ROLES    AGE     VERSION
ip-192-168-0-126.us-west-2.compute.internal    Ready    <none>   8m49s
 v1.22.9-eks-810597c
ip-192-168-0-92.us-west-2.compute.internal     Ready    <none>   8m34s
 v1.22.9-eks-810597c
```

b. Especifique em qual zona de disponibilidade cada nó se encontra. Execute o comando a seguir para cada nó, retornado na etapa anterior.

```
aws ec2 describe-instances --filters Name=network-interface.private-dns-
name,Values=ip-192-168-0-126.us-west-2.compute.internal \
--query 'Reservations[].Instances[].{AvailabilityZone:
 Placement.AvailabilityZone, SubnetId: SubnetId}'
```

Veja um exemplo de saída abaixo.

```
[
    {
        "AvailabilityZone": "us-west-2d",
        "SubnetId": "subnet-Example5"
    }
]
```

c. Anote cada nó com a `ENIConfig` que você criou para o ID da sub-rede e a zona de disponibilidade. Apenas é possível anotar um nó com uma `ENIConfig`, embora vários nós possam ser anotados com a mesma `ENIConfig`. Substitua *`example values`* pelos seus próprios valores.

```
kubectl annotate node ip-192-168-0-126.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName2
```

4. Se você tinhas nós em um cluster de produção com Pods em execução antes de mudar para utilizar o recurso de redes personalizadas, conclua as tarefas a seguir:

   a. Certifique-se de ter nós disponíveis que estão utilizando o recurso de rede personalizado.

   b. Cordon e drene os nós para desligar graciosamente Pods. Para obter mais informações, consulte Safely Drain a Node (Drenar um nó com segurança) na documentação do Kubernetes.

   c. Encerre os nós. Se os nós estiverem em um grupo de nós gerenciados existente, será possível excluir o grupo de nós. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado:

   - Substitua *`my-cluster`* pelo nome do seu cluster.
   - Substitua *`my-nodegroup`* pelo nome do seu grupo de nós.

```
aws eks delete-nodegroup --cluster-name my-cluster --nodegroup-name my-
nodegroup
```

Somente os novos nós que são registrados com o rótulo `k8s.amazonaws.com/eniConfig` usam o novo recurso de redes personalizadas.

5. Confirme que Pods recebem um endereço IP de um bloco CIDR associado a uma das sub-redes criadas em uma etapa anterior.

```
kubectl get pods -A -o wide
```

Veja um exemplo de saída abaixo.

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
           NODE                                            NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-2rkn4             1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   aws-node-k96wp             1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
kube-system   coredns-657694c6f4-smcgr   1/1     Running   0          56m
 192.168.1.23    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   coredns-657694c6f4-stwv9   1/1     Running   0          56m
 192.168.1.28    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-jgshq           1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-wx9vk           1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
```

É possível ver que os `coredns` Pods recebem endereços IP do bloco CIDR `192.168.1.0` que você adicionou à VPC. Sem redes personalizadas, eles teriam recebido endereços do bloco CIDR `192.168.0.0`, porque este era o único bloco CIDR originalmente associado à VPC.

Se um Pod's `spec` contiver `hostNetwork=true`, será atribuído o endereço IP primário do nó. Ele não receberá um endereço das sub-redes que foram adicionadas. Por padrão, esse valor é definido como `false`. Esse valor é definido como `true` para a `kube-proxy` e o Amazon VPC CNI plugin for Kubernetes (`aws-node`) Pods que são executados no seu cluster. É por esse motivo que não foram atribuídos endereços `192.168.1.`*x* ao `kube-proxy` e aos Pods do `aws-node` do plug-in na saída anterior. Para saber mais sobre a configuração `hostNetwork` do Pod's, consulte PodSpec v1 core na referência de APIs do Kubernetes.

Etapa 5: Excluir os recursos do tutorial

Depois de concluir o tutorial, convém excluir os recursos criados. Em seguida, ajuste as etapas para habilitar as redes personalizadas para um cluster de produção.

Para excluir os recursos do tutorial

1. Se o grupo de nós criado foi apenas para testes, exclua-o.

   ```
   aws eks delete-nodegroup --cluster-name $cluster_name --nodegroup-name my-nodegroup
   ```

   Mesmo após a saída de AWS CLI indicar que o cluster foi excluído, é possível que o processo de exclusão não esteja realmente concluído. O processo de exclusão demora alguns minutos. Confirme se a exclusão foi feita, executando o seguinte comando:

   ```
   aws eks describe-nodegroup --cluster-name $cluster_name --nodegroup-name my-
   nodegroup --query nodegroup.status --output text
   ```

   Não continue até que a saída retornada seja semelhante à seguinte:

   ```
   An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup
    operation: No node group found for name: my-nodegroup.
   ```

2. Se o grupo de nós criado foi apenas para testes, exclua a perfil do IAM do nó.

   a. Desanexe as políticas da função.

      ```
      aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
      policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
      ```

```
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

b. Exclua a função.

```
aws iam delete-role --role-name myCustomNetworkingAmazonEKSNodeRole
```

3. Excluir o cluster.

```
aws eks delete-cluster --name $cluster_name
```

Confirme se a exclusão do cluster foi concluída, com o comando a seguir.

```
aws eks describe-cluster --name $cluster_name --query cluster.status --output text
```

Quando uma saída semelhante à seguinte for retornada, significa que o cluster foi excluído com êxito:

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-cluster.
```

4. Exclua o perfil do IAM do cluster.

a. Desanexe as políticas da função.

```
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole
 --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
```

b. Exclua a função.

```
aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
```

5. Exclua as sub-redes que você criou em uma etapa anterior.

```
aws ec2 delete-subnet --subnet-id $new_subnet_id_1
aws ec2 delete-subnet --subnet-id $new_subnet_id_2
```

6. Exclua a pilha da VPC criada.

```
aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
```

Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2

Cada tipo de instância do Amazon EC2 oferece suporte a um número máximo de interfaces de rede elásticas e a um número máximo de endereços IP que podem ser atribuídos a cada interface de rede. Cada nó requer um endereço IP para cada interface de rede. Todos os outros endereços IP disponíveis podem ser atribuídos a `Pods`. Cada `Pod` exige seu próprio endereço IP. Como resultado, é possível ter nós com recursos de computação e memória disponíveis, mas que não são capazes de acomodar `Pods` adicionais porque o nó ficou sem endereços IP para atribuir aos `Pods`.

Neste tópico, você aprenderá a aumentar significativamente o número de endereços IP que os nós podem atribuir a `Pods` atribuindo prefixos IP em vez de atribuir endereços IP secundários individuais aos seus nós. Cada prefixo inclui vários endereços IP. Se você não configurar o cluster para atribuição de prefixos IP, seu cluster deverá fazer mais chamadas da interface de programação de aplicações (API) do Amazon EC2 para configurar as interfaces de rede e os endereços IP necessários para a conectividade de Pod. À medida que os clusters aumentam de tamanho, a frequência dessas chamadas de API podem levar a tempos de execução de instâncias e Pod maiores. Isso resulta em atrasos na escalabilidade para atender à demanda de workloads grandes e com alto pico, além de adicionar custos e despesas gerais de gerenciamento, pois você precisará provisionar clusters e VPCs adicionais para atender aos requisitos de escalabilidade. Para obter mais informações, consulte Limites de escalabilidade do Kubernetes no GitHub.

Considerações

- Cada instância do Amazon EC2 oferece suporte um número máximo de Pods. Se o grupo de nós gerenciados consistir em vários tipos de instância, o menor número de Pods máximos para uma instância no cluster é aplicado a todos os nós do cluster.
- Por padrão, o número máximo de `Pods` que podem ser executados em um nó é 110, mas esse número pode ser alterado. Se você alterar o número e tiver um grupo de nós gerenciados existente, a próxima atualização da AMI ou do modelo de execução do seu grupo de nós resultará no surgimento de novos nós com o valor alterado.
- Ao fazer a transição da atribuição de endereços IP para a atribuição de prefixos IP, recomendamos criar novos grupos de nós para aumentar o número de endereços IP disponíveis, em vez de fazer uma substituição contínua dos nós existentes. A execução de Pods em um nó com endereços IP e prefixos atribuídos pode causar inconsistência na capacidade de endereço IP anunciada, afetando

as futuras workloads no nó. Para saber a forma recomendada de realizar a transição, consulte Substituir todos os nós durante a migração do modo de IP secundário para o modo de delegação de prefixo ou vice-versa no guia de práticas recomendadas do Amazon EKS.

- Para clusters somente com nós Linux.
  - Depois de configurar o complemento para atribuir prefixos a interfaces de rede, você não poderá fazer downgrade do complemento Amazon VPC CNI plugin for Kubernetes para uma versão inferior à `1.9.0` (ou `1.10.1`) sem remover todos os nós em todos os grupos de nós do cluster.
  - Se você usa grupos de segurança para Pods, com `POD_SECURITY_GROUP_ENFORCING_MODE = standard` e `AWS_VPC_K8S_CNI_EXTERNALSNAT =false`, quando seus Pods se comunicarem com endpoints fora da VPC, os grupos de segurança do nó serão usados, em vez de quaisquer grupos de segurança que você possa ter atribuído a seus Pods.

    Se você também estiver usando grupos de segurança para Pods, com `POD_SECURITY_GROUP_ENFORCING_MODE =strict`, quando seus `Pods` se comunicarem com endpoints fora da sua VPC, os grupos de segurança do `Pod's` serão usados.

Pré-requisitos

- Um cluster existente. Para implantar, consulte Criar um cluster do Amazon EKS..
- As sub-redes nas quais seus nós do Amazon EKS estão devem ter blocos de Encaminhamento Entre Domínios Sem Classificação (CIDR) `/28` (para clusters `IPv4`) ou `/80` (para clusters `IPv6`) suficientemente contíguos. Somente nós Linux podem existir em um cluster `IPv6`. O uso de prefixos IP poderá falhar se os endereços IP estiverem espalhados pelo CIDR da sub-rede. Recomendamos o seguinte:
  - Usar uma reserva CIDR de sub-rede para que, mesmo que algum endereço IP dentro do intervalo reservado ainda esteja em uso, após sua liberação, os endereços IP não sejam reatribuídos. Isso garante que os prefixos estejam disponíveis para alocação sem segmentação.
  - Usar novas sub-redes que sejam utilizadas especificamente para executar workloads às quais os prefixos IP estão atribuídos. Tanto as workloads Windows quanto as workloads Linux podem ser executadas na mesma sub-rede ao atribuir prefixos IP.
- Para atribuir prefixos IP aos seus nós, eles devem ser baseados no AWS Nitro. As instâncias que não são baseadas em Nitro continuam a alocar endereços IP secundários individuais, mas têm um número significativamente menor de endereços IP a atribuir aos Pods do que as instâncias Nitro-based.

- Somente para clusters com nós do Linux: se o cluster estiver configurado para a família `IPv4`, você deverá ter a versão `1.9.0` ou posterior do complemento Amazon VPC CNI plugin for Kubernetes instalada. É possível verificar a versão atual com o comando a seguir.

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/"
 -f 2
```

  Se o cluster estiver configurado para a família `IPv6`, a versão `1.10.1` ou posterior do complemento deverá estar instalada. Se a versão do plug-in for anterior às versões exigidas, será necessário atualizá-lo. Para mais informações, consulte as seções de atualização de Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

- Para clusters somente com nós do Windows
  - Seu cluster e a respectiva versão da plataforma devem ser iguais ou posteriores às versões na tabela a seguir. Para atualizar a versão do cluster, consulte Atualizar um cluster existente para a nova versão do Kubernetes. Se o seu cluster não estiver na versão mínima da plataforma, não será possível atribuir prefixos IP aos seus nós até que o Amazon EKS atualize a versão da sua plataforma.

Versão do Kubernetes	Versão da plataforma
`1.27`	`eks.3`
`1.26`	`eks.4`
`1.25`	`eks.5`

  É possível verificar a versão atual do Kubernetes e da plataforma substituindo *`my-cluster`* no comando a seguir pelo nome do seu cluster e, em seguida, executando o comando modificado: **`aws eks describe-cluster --name my-cluster --query 'cluster.{"Kubernetes Version": version, "Platform Version": platformVersion}'`**.

  - Suporte ao Windows habilitado para seu cluster. Para ter mais informações, consulte Implantar nós do Windows em clusters do EKS.

Para aumentar a quantidade de endereços IP disponíveis para seus nós do Amazon EC2

1. Configure seu cluster para atribuir prefixos de endereço IP aos nós. Conclua o procedimento na guia correspondente ao sistema operacional do seu nó.

   Linux

   1. Habilite o parâmetro para atribuir prefixos a interfaces de rede para o `DaemonSet` CNI da Amazon VPC. Ao implantar um cluster da versão `1.21` ou superior, a versão `1.10.1` ou superior do complemento Amazon VPC CNI plugin for Kubernetes é implantada com ele. Se você criou o cluster com a família `IPv6`, essa configuração foi definida como `true` por padrão. Se você criou o cluster com a família `IPv4`, essa configuração foi definida como `false` por padrão.

      ```
      kubectl set env daemonset aws-node -n kube-system
       ENABLE_PREFIX_DELEGATION=true
      ```

      > **Important**
      >
      > Mesmo que sua sub-rede tenha endereços IP disponíveis, se ela não tiver blocos `/28` contíguos disponíveis, você verá o erro a seguir nos logs do Amazon VPC CNI plugin for Kubernetes.
      >
      > ```
      > InsufficientCidrBlocks: The specified subnet does not have enough free
       cidr blocks to satisfy the request
      > ```
      >
      > Isso pode acontecer devido à fragmentação dos endereços IP secundários existentes espalhados por uma sub-rede. Para resolver esse erro, crie uma nova sub-rede e inicie Pods lá ou use uma reserva CIDR de sub-rede do Amazon EC2 para reservar espaço em uma sub-rede para uso com atribuição de prefixo. Para obter mais informações, consulte Comportamento do endereçamento IP para sua sub-rede no Guia do usuário da Amazon VPC.

   2. Se você planeja implantar um grupo de nós gerenciados sem um modelo de inicialização, ou com um modelo de inicialização no qual você não especificou um ID de AMI, e estiver usando uma versão do Amazon VPC CNI plugin for Kubernetes equivalente ou superior às versões listadas nos pré-requisitos, avance para a próxima etapa. Os grupos de nós gerenciados calculam automaticamente o número máximo de Pods para você.

Se você estiver implantando um grupo de nós autogerenciado ou um grupo de nós gerenciado com um modelo de lançamento no qual você especificou um ID de AMI, será necessário determinar o número recomendado do Amazon EKS de Pods máximos para seus nós. Siga as instruções em Máximo recomendado de Pods do Amazon EKS para cada tipo de instância do Amazon EC2, adicionando **`--cni-prefix-delegation-enabled`** na etapa 3. Anote a saída para uso em uma etapa superior.

> **Important**
>
> Grupos de nós gerenciados impõem um número máximo para o valor de `maxPods`. Para instâncias com menos de 30 vCPUs, o número máximo é 110 e, para todas as outras instâncias, o número máximo é 250. Esse número máximo é aplicado quer a delegação de prefixo esteja habilitada ou não.

3. Se você estiver usando um cluster versão `1.21` ou superior configurado para `IPv6`, pule para a próxima etapa.

   Especifique os parâmetros em uma das opções a seguir. Para determinar qual é opção certa para você e qual é o valor a ser fornecido para ela, consulte `WARM_PREFIX_TARGET`, `WARM_IP_TARGET` e `MINIMUM_IP_TARGET` no GitHub.

   Você pode substituir os *`example values`* por um valor maior que zero.

   - `WARM_PREFIX_TARGET`

     ```
     kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
     ```

   - `WARM_IP_TARGET` ou `MINIMUM_IP_TARGET` – Se qualquer um dos dois valores for definido, substitui qualquer valor definido para `WARM_PREFIX_TARGET`.

     ```
     kubectl set env ds aws-node -n kube-system WARM_IP_TARGET=5
     ```

     ```
     kubectl set env ds aws-node -n kube-system MINIMUM_IP_TARGET=2
     ```

4. Crie um dos seguintes tipos de grupos de nós com pelo menos um tipo de instância do Amazon EC2 Nitro Amazon Linux 2. Para obter uma lista dos tipos de instância do Nitro, consulte Instâncias criadas no Sistema Nitro no Guia do usuário do Amazon EC2. Este

recurso não é compatível com o Windows. Para as opções que incluem *110*, substitua pelo valor da etapa 3 (recomendado) ou por seu próprio valor.

- Autogerenciado: cria o grupo de nós usando as instruções em Criar nós autogerenciados do Amazon Linux. Especifique o seguinte texto para o parâmetro BootstrapArguments.

```
--use-max-pods false --kubelet-extra-args '--max-pods=110'
```

Se estiver usando `eksctl` para criar o grupo de nós, poderá usar o comando a seguir.

```
eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-
node 110
```

- Managed (Gerenciado) – Implante seu grupo de nós usando uma das seguintes opções:
  - Sem um modelo de lançamento ou com um modelo de lançamento sem um ID de AMI especificado – Conclua o procedimento em Criar um grupo de nós gerenciados para seu cluster. Grupos de nós gerenciados agora calculam automaticamente o valor recomendado de `max-pods` do Amazon EKS para você.
  - Com um modelo de lançamento com um ID de AMI especificado – Em seu modelo de lançamento, especifique um ID de AMI otimizado do Amazon EKS ou uma AMI personalizada criada a partir da AMI otimizada do Amazon EKS e, em seguida implante o grupo de nós usando um modelo de lançamento e forneça os seguintes dados do usuário no modelo de lançamento. Esses dados do usuário transmitem argumentos para o arquivo `bootstrap.sh`. Para obter mais informações sobre o arquivo de bootstrap, consulte bootstrap.sh no GitHub.

```
/etc/eks/bootstrap.sh my-cluster \
  --use-max-pods false \
  --kubelet-extra-args '--max-pods=110'
```

Se estiver usando `eksctl` para criar o grupo de nós, poderá usar o comando a seguir.

```
eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
```

Se você criou uma AMI personalizada que não foi criada a partir da AMI otimizada do Amazon EKS, precisa criar a configuração você mesmo.

> **Note**
>
> Se você também quiser atribuir endereços IP aos Pods de uma sub-rede diferente da sub-rede da instância, será necessário habilitar o recurso nesta etapa. Para ter mais informações, consulte Rede personalizada para pods.

Windows

1. Habilite a atribuição de prefixos IP.
   a. Abra o `ConfigMap` de `amazon-vpc-cni` para edição.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. Adicione as linhas a seguir à seção `data`:

   ```
     enable-windows-prefix-delegation: "true"
   ```

   c. Salve o arquivo e feche o editor.
   d. Confirme se a anotação foi adicionada ao `ConfigMap`.

   ```
   kubectl get configmap -n kube-system amazon-vpc-cni -o
    "jsonpath={.data.enable-windows-prefix-delegation}"
   ```

   Se a saída retornada não for `true`, é possível que um erro tenha ocorrido. Tente concluir a etapa novamente.

   > **Important**
   >
   > Mesmo que sua sub-rede tenha endereços IP disponíveis, se ela não tiver blocos `/28` contíguos disponíveis, você verá o erro a seguir nos eventos do nó.
   >
   > ```
   > "failed to allocate a private IP/Prefix address:
   >  InsufficientCidrBlocks: The specified subnet does not have enough
   >  free cidr blocks to satisfy the request"
   > ```

Isso pode acontecer devido à fragmentação dos endereços IP secundários existentes espalhados por uma sub-rede. Para resolver esse erro, crie uma nova sub-rede e inicie Pods lá ou use uma reserva CIDR de sub-rede do Amazon EC2 para reservar espaço em uma sub-rede para uso com atribuição de prefixo. Para obter mais informações, consulte Comportamento do endereçamento IP para sua sub-rede no Guia do usuário da Amazon VPC.

2. (Opcional) Especifique uma configuração adicional para controlar o comportamento de pré-escalabilidade e escalabilidade dinâmica do seu cluster. Para obter mais informações, consulte Opções de configuração com o modo de delegação de prefixo ativado Windows no GitHub.

   a. Abra o `ConfigMap` de `amazon-vpc-cni` para edição.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. Substitua *`example values`* por um valor maior que zero e adicione as entradas necessárias à seção `data` do `ConfigMap`. Se você definir um valor para um `warm-ip-target` ou `minimum-ip-target`, o valor substituirá qualquer valor definido para `warm-prefix-target`.

   ```
   warm-prefix-target: "1"
   warm-ip-target: "5"
   minimum-ip-target: "2"
   ```

   c. Salve o arquivo e feche o editor.

3. Crie grupos de nós Windows com pelo menos um tipo de instância do Nitro do Amazon EC2. Para obter uma lista dos tipos de instância do Nitro, consulte Instâncias criadas no sistema Nitro no Guia do usuário do Amazon EC2. Por padrão, o número máximo de Pods podem ser implantados em um nó é 110. Se você quiser aumentar ou diminuir esse número, especifique as informações a seguir nos dados do usuário para a configuração do bootstrap. Substitua *`max-pods-quantity`* pelos seus valores máximos de pods.

   ```
   -KubeletExtraArgs '--max-pods=max-pods-quantity'
   ```

   Se você estiver implantando grupos de nós gerenciados, essa configuração precisará ser adicionada ao modelo de inicialização. Para ter mais informações, consulte Personalizar

nós gerenciados com modelos de execução. Para obter mais informações sobre os parâmetros de configuração do script de bootstrap do Windows, consulte Parâmetros de configuração do script de bootstrap.

2. Depois que seus nós forem implantados, visualize-os no cluster.

```
kubectl get nodes
```

Veja um exemplo de saída abaixo.

```
NAME                                             STATUS     ROLES    AGE   VERSION
ip-192-168-22-103.region-code.compute.internal   Ready      <none>   19m   v1.XX.X-eks-6b7464
ip-192-168-97-94.region-code.compute.internal    Ready      <none>   19m   v1.XX.X-eks-6b7464
```

3. Descreva um dos nós para determinar o valor de `max-pods` para o nó e o número de endereços IP disponíveis. Substitua *192.168.30.193* pelo endereço `IPv4` no nome de um dos nós retornados na saída da etapa anterior.

```
kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|PrivateIPv4Address'
```

Veja um exemplo de saída abaixo.

```
pods:                                       110
vpc.amazonaws.com/PrivateIPv4Address:  144
```

Na saída anterior, `110` é o número máximo de Pods que o Kubernetes implantará no nó, mesmo que *144* endereços IP estejam disponíveis.

Grupos de segurança do Pods

Os grupos de segurança para Pods integram os grupos de segurança do Amazon EC2 com os Pods do Kubernetes. Você pode usar grupos de segurança do Amazon EC2 para definir regras que permitem tráfego de rede de entrada e saída de e para Pods implantados em nós executados em muitos tipos de instância do Amazon EC2 e Fargate. Para obter uma explicação detalhada sobre esse recurso, consulte o post do blog Introducing grupo de segurança for Pods (Apresentação de grupos de segurança para pods).

Considerações

- Antes de implantar grupos de segurança para Pods, considere as seguintes limitações e condições:
- Os grupos de segurança para Pods não podem ser usados com nós do Windows.
- Os grupos de segurança para Pods podem ser usados com clusters configurados para a família `IPv6` que contém nós do Amazon EC2 usando a versão 1.16.0 ou posterior do plug-in CNI do Amazon VPC. Você pode usar grupos de segurança para `IPv6` com clusters configurados para a família Pods que contêm apenas nós Fargate usando a versão 1.7.7 ou posterior do plug-in CNI do Amazon VPC. Para ter mais informações, consulte Endereços `IPv6` para clusters, Pods e services.
- Os grupos de segurança para os Pods são compatíveis com a maioria das famílias de instâncias do Amazon EC2 baseadas no Nitro, mas não com todas as gerações de uma família. Por exemplo, eles são compatíveis com a família e as gerações de instâncias `m5`, `c5`, `r5`, `m6g`, `c6g` e `r6g`. Mas não são compatíveis com nenhum tipo de instância da família `t`. Para obter uma lista completa de tipos de instância compatíveis, consulte o arquivo limits.go no GitHub. Seus nós devem ser um dos tipos de instâncias listados que têm `IsTrunkingCompatible: true` nesse arquivo.
- Se você também estiver usando políticas de segurança de Pod para restringir o acesso à mutação de Pods, o usuário do `eks:vpc-resource-controller` Kubernetes deverá ser especificado no `ClusterRoleBinding` do Kubernetes para o `role` ao qual o `psp` está atribuído. Se você estiver usando o Amazon EKS padrão `psp`, `role` e `ClusterRoleBinding`, este é o `eks:podsecuritypolicy:authenticated` `ClusterRoleBinding`. Por exemplo, você adiciona o usuário à seção `subjects:`, como mostrado no seguinte exemplo:

```
[...]
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: system:authenticated
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: eks:vpc-resource-controller
  - kind: ServiceAccount
    name: eks-vpc-resource-controller
```

- Se você estiver usando redes personalizadas e grupos de segurança para Pods juntos, o grupo de segurança especificado por grupos de segurança para Pods é usado em vez do grupo de segurança especificado no `ENIConfig`.

- Se você estiver utilizando a versão `1.10.2` ou anterior do plug-in Amazon VPC CNI e incluir a configuração `terminationGracePeriodSeconds` na especificação do Pod, o valor dessa configuração não poderá ser zero.
- Se estiver utilizando a versão `1.10` ou anterior do plug-in CNI da Amazon VPC ou a versão `1.11` com `POD_SECURITY_GROUP_ENFORCING_MODE` = `strict`, que é a configuração padrão, os serviços do Kubernetes do tipo `NodePort` e `LoadBalancer` que usam destinos de instância com a `externalTrafficPolicy` definida como `Local` não serão compatíveis com os Pods a que você atribuir grupos de segurança. Para obter mais informações sobre como usar um balanceador de carga com destinos de instância, consulte Roteamento TCP e tráfego UDP com Network Load Balancers.
- Se estiver utilizando a versão `1.10` ou anterior do plugin Amazon VPC CNI ou a versão `1.11` com `POD_SECURITY_GROUP_ENFORCING_MODE=strict`, que é a configuração padrão, o NAT de origem estará desabilitado para tráfego de saída de Pods com grupos de segurança atribuídos para que as regras de grupo de segurança de saída sejam aplicadas. Para acessar a Internet, Pods com grupos de segurança atribuídos devem ser iniciados em nós implantados em uma sub-rede privada configurada com um gateway ou instância NAT. Pods com grupos de segurança atribuídos implantados em sub-redes públicas não podem acessar a Internet.

  Se estiver utilizando a versão `1.11` ou posterior do plugin com `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, o tráfego do Pod destinado para fora da VPC será convertido no endereço IP da interface de rede primária da instância. Para esse tráfego, as regras nos grupos de segurança da interface de rede principal são utilizadas, e não as regras nos grupos de segurança do Pod's.
- Para usar a política de rede do Calico com Pods que têm grupos de segurança associados, é necessário usar a versão `1.11.0` ou posterior do plug-in Amazon VPC CNI e definir `POD_SECURITY_GROUP_ENFORCING_MODE=standard`. Caso contrário, o fluxo de tráfego entrando e saindo de Pods com grupos de segurança associados não estará sujeito à política de rede do Calico e estará limitado apenas à imposição do grupo de segurança do Amazon EC2. Para atualizar sua versão do Amazon VPC CNI, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.
- Pods em execução em nós do Amazon EC2 que utilizam grupos de segurança em clusters que utilizam Nodelocal DNSCache apenas têm suporte com a versão `1.11.0` ou posterior do plugin Amazon VPC CNI e com `POD_SECURITY_GROUP_ENFORCING_MODE=standard`. Para atualizar sua versão do plugin Amazon VPC CNI, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

- Grupos de segurança para Pods podem levar a uma latência de inicialização mais alta Pod no para o Pods com alta rotatividade. Isso se deve à limitação de taxa no controlador de recursos.

Configurar o Amazon VPC CNI plugin for Kubernetes para grupos de segurança de Pods

Para implantar grupos de segurança para Pods

Se você estiver usando grupos de segurança somente para Pods Fargate e não tiver nenhum nó do Amazon EC2 em seu cluster, pule para a Implantar uma aplicação de exemplo.

1. Verifique a versão atual de Amazon VPC CNI plugin for Kubernetes com o comando a seguir:

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

   Veja um exemplo de saída abaixo.

```
v1.7.6
```

   Se a versão do Amazon VPC CNI plugin for Kubernetes for anterior à `1.7.7`, atualize o plugin para a versão `1.7.7` ou posterior. Para ter mais informações, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

2. Adicione a política gerenciada do IAM `AmazonEKSVPCResourceController` à função do cluster que estiver associada ao cluster do Amazon EKS. A política permite que a função gerencie interfaces de rede, seus endereços IP privados e seu anexo e desprendimento de e para instâncias de rede.

   a. Recupere o nome do perfil do IAM do cluster e armazene-o em uma variável. Substitua o *`my-cluster`* pelo nome do cluster.

```
cluster_role=$(aws eks describe-cluster --name my-cluster --query
 cluster.roleArn --output text | cut -d / -f 2)
```

   b. Anexe a política ao perfil.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSVPCResourceController --role-name $cluster_role
```

3. Habilite o complemento Amazon VPC CNI para gerenciar interfaces de rede para Pods, definindo a variável `ENABLE_POD_ENI` como `true` no `aws-node`DaemonSet Uma vez que essa configuração é definida como `true`, para cada nó no cluster, o complemento cria um recurso personalizado cninode. O controlador de recursos da VPC cria e anexa uma interface de rede especial chamada Interface de rede com a descrição `aws-k8s-trunk-eni`.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
```

> **Note**
>
> A interface de rede de tronco está incluída no número máximo de interfaces de rede suportadas pelo tipo de instância. Para obter uma lista do número máximo de interfaces de rede com suporte por cada tipo de instância, consulte IP addresses per network interface per instance type (Endereços IP por interface de rede por tipo de instância) no Guia do usuário do Amazon EC2. Se o nó já tiver o número máximo de interfaces de rede padrão anexadas a ele, o controlador de recursos da VPC reservará um espaço. Você terá que reduzir seus Pods de execução o suficiente para que o controlador desconecte e exclua uma interface de rede padrão, crie a interface de rede de tronco e anexe-a à instância.

4. Você pode ver quais de seus nós possuem um recurso personalizado do `CNINode` com o comando a seguir. Se `No resources found` for retornado, aguarde alguns segundos e tente de novo. A etapa anterior requer a reinicialização dos Amazon VPC CNI plugin for de KubernetesPods, o que demora vários segundos.

```
$ kubectl get cninode -A
     NAME FEATURES
     ip-192-168-64-141.us-west-2.compute.internal
 [{"name":"SecurityGroupsForPods"}]
     ip-192-168-7-203.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
```

Se você estiver usando versões do VPC CNI anteriores a `1.15`, rótulos de nós foram usados em vez do recurso `CNINode` personalizado. Você pode ver quais de seus nós têm o rótulo de nó `aws-k8s-trunk-eni` definido para `true` com o comando a seguir. Se `No resources found` for retornado, aguarde alguns segundos e tente de novo. A etapa anterior requer a reinicialização dos Amazon VPC CNI plugin for de KubernetesPods, o que demora vários segundos.

```
kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
-
```

Depois que a interface de rede de tronco for criada, os Pods receberão endereços IP secundários a partir das interfaces de tronco ou de rede padrão. A interface de tronco é excluída automaticamente se o nó for excluído.

Quando você implanta um grupo de segurança para um Pod em uma etapa superior, o controlador de recursos da VPC cria uma interface de rede especial chamada interface de rede da ramificação, com uma descrição de `aws-k8s-branch-eni` e associa os grupos de segurança a ela. As interfaces de rede da ramificação são criadas como um adicional para as interfaces de rede padrão e de tronco conectadas ao nó.

Se você estiver usando testes de liveness ou prontidão, também precisará desabilitar o TCP early demux, de modo que o `kubelet` possa se conectar a Pods em interfaces de rede de ramificações usando TCP. Para desativar o TCP early demux, execute o seguinte comando:

```
kubectl patch daemonset aws-node -n kube-system \
  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":
[{"name":"DISABLE_TCP_EARLY_DEMUX","value":"true"}],"name":"aws-vpc-cni-
init"}]}}}}'
```

> **Note**
>
> Se estiver usando a versão `1.11.0` ou posterior do complemento Amazon VPC CNI plugin for Kubernetes e definir `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, conforme descrito na próxima etapa, não será necessário executar o comando anterior.

5. Se o cluster usar `NodeLocal DNSCache` ou se você desejar usar a política de rede do Calico com Pods que têm seus próprios grupos de segurança ou se tiver serviços do Kubernetes do tipo `NodePort` e `LoadBalancer` que usam destinos de instâncias com um `externalTrafficPolicy` definido como `Local` para Pods aos quais deseja atribuir grupos de segurança, você deverá estar usando a versão `1.11.0` ou posterior do complemento Amazon VPC CNI plugin for Kubernetes e habilitar a seguinte configuração:

```
kubectl set env daemonset aws-node -n kube-system
 POD_SECURITY_GROUP_ENFORCING_MODE=standard
```

> **Important**
>
> - Regras de grupos de segurança de Pod não são aplicadas ao tráfego entre Pods ou Pods e services, por exemplo, `kubelet` ou `nodeLocalDNS`, que estejam no mesmo nó. Pods que usam diferentes grupos de segurança no mesmo nó não conseguem se comunicar porque estão configurados em sub-redes diferentes e o roteamento está desativado entre essas sub-redes.
> - O tráfego de saída de Pods para endereços fora da VPC é o endereço de rede convertido no endereço IP da interface de rede primária da instância (a menos que você também tenha definido `AWS_VPC_K8S_CNI_EXTERNALSNAT=true`). Para esse tráfego, as regras nos grupos de segurança da interface de rede principal são utilizadas, e não as regras nos grupos de segurança do Pod's.
> - Para que essa configuração seja aplicada a Pods existentes, é necessário reiniciar osPods ou os nós que nos quais esses Pods estão sendo executados.

Implantar uma aplicação de exemplo

Para utilizar grupos de segurança para Pods, é necessário ter um grupo de segurança existente e Implantar um `SecurityGroupPolicy` do Amazon EKS no cluster, conforme descrito no procedimento a seguir. As etapas seguintes mostram como utilizar a política de grupo de segurança para um Pod: A menos que indicado de outra forma, conclua todas as etapas no mesmo terminal, pois são utilizadas nas etapas a seguir variáveis que não persistem entre terminais.

Para implantar um Pod de exemplo com um grupo de segurança

1. Crie um namespace do Kubernetes no qual implantar recursos. Você pode substituir *my-namespace* pelo nome de um namespace que deseje usar.

   ```
   kubectl create namespace my-namespace
   ```

2. Implante uma `SecurityGroupPolicy` do Amazon EKS no cluster.

   a. Copie o conteúdo a seguir para o seu dispositivo. Você pode substituir *podSelector* por **serviceAccountSelector** se preferir selecionar Pods com base em rótulos de conta de serviço. É preciso especificar um seletor ou outro. Uma `podSelector` vazia (exemplo: `podSelector: {}`) seleciona todos os Pods no namespace. Você pode alterar *my-role*

para o nome do seu perfil. Uma `serviceAccountSelector` vazia seleciona todas as contas de serviço no namespace. Você pode substituir *`my-security-group-policy`* por um nome para a `SecurityGroupPolicy` e *`my-namespace`* pelo namespace no qual você deseja criar a `SecurityGroupPolicy`.

Você deve substituir *`my_pod_security_group_id`* pelo ID de um grupo de segurança existente. Se você ainda não tiver um grupo de segurança, deverá criar um. Para obter mais informações, consulte Grupos de segurança do Amazon EC2 para instâncias do Linux no Guia do usuário do Amazon EC2. Você deve especificar de 1 a 5 IDs de grupo de segurança. Se você especificar mais de um ID, a combinação de todas as regras em todos os grupos de segurança será efetiva para os Pods selecionados.

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - my_pod_security_group_id
EOF
```

Important

Os grupos de segurança que você especificar para seus Pods devem atender aos seguintes critérios:

- Eles devem existir. Se eles não existirem, então, quando você implantar um Pod que corresponda ao seletor, o Pod permanecerá preso no processo de criação. Se você descrever o Pod, verá uma mensagem de erro semelhante à seguinte: `An error occurred (InvalidSecurityGroupID.NotFound) when calling the CreateNetworkInterface operation: The securityGroup ID 'sg-05b1d815d1EXAMPLE' does not exist.`

- Eles devem permitir comunicação de entrada do grupo de segurança aplicado aos nós (para `kubelet`) por todas as portas para as quais você configurou sondagens.
- Eles devem permitir comunicação de saída pelas portas TCP e UDP 53 com um grupo de segurança atribuído aos Pods (ou nós em que os Pods são executados) sendo executados no CoreDNS. O grupo de segurança dos Pods CoreDNS devem permitir tráfego de entrada na porta TCP e UDP 53 do grupo de segurança que você especificar.
- Eles devem ter regras de entrada e saída necessárias para comunicação com outros Pods com os quais precisam se comunicar.
- Eles devem ter regras que permitam que os Pods se comuniquem com o ambiente de gerenciamento do Kubernetes se você estiver usando esse grupo de segurança com o Fargate. A maneira mais fácil de fazer isso é especificar o grupo de segurança de cluster como um dos grupos de segurança.

As políticas de grupo de segurança só se aplicam a pods recém-agendados Pods. Eles não afetam os Pods em execução.

b. Implante a política.

```
kubectl apply -f my-security-group-policy.yaml
```

3. Implante um exemplo de aplicação com um rótulo que corresponda ao valor de *my-role* para o *podSelector* que você especificou na etapa anterior.

a. Copie o conteúdo a seguir para o seu dispositivo. Substitua os *exemplos de valor* pelos seus próprios e execute o comando modificado. Se você substituir *my-role*, certifique-se de que seja o mesmo valor que você especificou para o seletor em uma etapa anterior.

```
cat >sample-application.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
```

```
  replicas: 4
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
        role: my-role
    spec:
      terminationGracePeriodSeconds: 120
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
EOF
```

b. Implante a aplicação com o comando a seguir. Quando você implanta a aplicação, Amazon VPC CNI plugin for Kubernetes corresponde à etiqueta `role` e os grupos de segurança especificados na etapa anterior são aplicados ao Pod.

```
kubectl apply -f sample-application.yaml
```

4. Visualizar os Pods Implantados com a aplicação de amostra. Para o restante do tópico, esse terminal é indicado como `TerminalA`.

```
kubectl get pods -n my-namespace -o wide
```

Veja um exemplo de saída abaixo.

```
NAME                              READY   STATUS    RESTARTS   AGE     IP
       NODE                                            NOMINATED NODE   READINESS
 GATES
my-deployment-5df6f7687b-4fbjm   1/1     Running   0          7m51s   192.168.53.48
    ip-192-168-33-28.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-j9fl4   1/1     Running   0          7m51s
  192.168.70.145   ip-192-168-92-33.region-code.compute.internal   <none>
 <none>
my-deployment-5df6f7687b-rjxcz   1/1     Running   0          7m51s
  192.168.73.207   ip-192-168-92-33.region-code.compute.internal   <none>
 <none>
my-deployment-5df6f7687b-zmb42   1/1     Running   0          7m51s   192.168.63.27
    ip-192-168-33-28.region-code.compute.internal   <none>           <none>
```

> **Note**
>
> - Se houver Pods travados no estado `Waiting`, execute `kubectl describe pod` ***my-deployment-xxxxxxxxxx-xxxxx*** `-n` *my-namespace*. Se você observar `Insufficient permissions: Unable to create Elastic Network Interface.`, confirme se adicionou a política do IAM à função de cluster do IAM em uma etapa anterior.
> - Se houver algum Pods preso no estado `Pending`, confirme se o tipo de instância do nó está listado em `limits.go` e se o número máximo de interfaces de rede de ramificação aceito pelo o tipo de instância multiplicado pelo número de nós do grupo de nós ainda não foi atingido. Por exemplo, um `m5.large` suporta nove interfaces de rede de ramificação. Se o grupo de nós tiver cinco nós, um máximo de 45 interfaces de rede de ramificação poderá ser criado para o grupo de nós. O 46º Pod que você tentar implantar ficará no estado `Pending` até que outro Pod que tenha grupos de segurança associados seja excluído.

Se você executar `kubectl describe pod` ***my-deployment-xxxxxxxxxx-xxxxx*** **-n** ***my-namespace*** e ver uma mensagem semelhante à seguinte, ela pode ser ignorada com

segurança: Essa mensagem pode aparecer quando o Amazon VPC CNI plugin for Kubernetes tenta configurar a rede de host e falha enquanto a interface de rede está sendo criada. O plugin CNI registra esse evento até que a interface de rede seja criada.

```
Failed to create Pod sandbox: rpc error: code = Unknown desc = failed to set up
 sandbox container
 "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f" network for Pod
 "my-deployment-5df6f7687b-4fbjm": networkPlugin
cni failed to set up Pod "my-deployment-5df6f7687b-4fbjm-c89wx_my-namespace"
 network: add cmd: failed to assign an IP address to container
```

Você não pode exceder o número máximo de Pods que podem ser executados no tipo de instância. Para obter uma lista do número máximo de Pods que você pode executar em cada tipo de instância, consulte eni-max-pods.txt no GitHub. Quando você exclui um Pod que tenha grupos de segurança associados ou exclui o nó em que o Pod está sendo executado, o controlador de recursos da VPC exclui a interface de rede da ramificação. Se você excluir um cluster com Pods usando Pods para grupos de segurança, o controlador não excluirá as interfaces de rede da ramificação, portanto, você precisará excluí-las você mesmo. Para obter mais informações sobre como excluir interfaces de rede, consulte Interfaces de rede elásticas no Guia do usuário do Amazon EC2.

5. Em um terminal à parte, faça shell em um dos Pods. Para o restante do tópico, esse terminal é indicado como `TerminalB`. Substitua *`5df6f7687b-4fbjm`* pelo ID de um dos Pods retornados na saída da etapa anterior.

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

6. No shell em `TerminalB`, confirme se a aplicação de amostra funciona.

```
curl my-app
```

Veja um exemplo de saída abaixo.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

Você recebeu a saída porque todos Pods que executam a aplicação estão associados ao grupo de segurança que você criou. Esse grupo contém uma regra que permite todo o tráfego entre todos Pods aos quais o grupo de segurança está associado. O tráfego DNS pode sair desse grupo de segurança para o grupo de segurança do cluster, que está associado aos seus nós. Os nós estão executando os Pods CoreDNS, para os quais seus Pods fizeram uma pesquisa de nome.

7. No `TerminalA`, remova as regras de grupo de segurança que permitem a comunicação DNS com o grupo de segurança de cluster do grupo de segurança. Se você não tiver adicionado as regras de DNS ao grupo de segurança de cluster em uma etapa anterior, substitua *`$my_cluster_security_group_id`* pelo ID do grupo de segurança no qual as regras foram criadas.

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_tcp_rule_id
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_udp_rule_id
```

8. No `TerminalB`, tente acessar a aplicação novamente.

```
curl my-app
```

Veja um exemplo de saída abaixo.

```
curl: (6) Could not resolve host: my-app
```

A tentativa falha porque o Pod não é mais capaz de acessar os Pods CoreDNS, que têm o grupo de segurança de cluster associado. O grupo de segurança do cluster não tem mais as regras de grupo de segurança que permitem a comunicação DNS do grupo de segurança associado ao Pod.

Se você tentar acessar a aplicação utilizando os endereços IP retornados para um dos Pods em uma etapa anterior, ainda receberá uma resposta, pois todas as portas são permitidas entre Pods que têm o grupo de segurança associado a eles, e uma pesquisa de nome não é necessária.

9. Quando você terminar os testes, poderá remover o exemplo de política de grupo de segurança, a aplicação e o grupo de segurança criados. Execute os comandos a seguir no `TerminalA`.

```
kubectl delete namespace my-namespace
aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
security-group-rule-ids $my_inbound_self_rule_id
wait
sleep 45s
aws ec2 delete-security-group --group-id $my_pod_security_group_id
```

Várias interfaces de rede para Pods

O Multus CNI é um plugin de interface de rede de contêiner (CNI) para o Amazon EKS que permite anexar várias interfaces de rede a um Pod. Para obter mais informações, consulte a documentação do Multus-CNI no GitHub.

No Amazon EKS, cada Pod tem uma interface de rede atribuída pelo plugin Amazon VPC CNI. Com o Multus, você pode criar um Pod multi-homes com várias interfaces. Isso é feito com o Multus atuando como um "meta-plugin", um plugin de CNI que pode chamar vários outros plugins de CNI. O suporte da AWS ao Multus vem configurado com o plugin Amazon VPC CNI como plugin delegado padrão.

Considerações

- O Amazon EKS não vai criar nem publicar plugins de CNI de virtualização de E/S raiz (SR-IOV) e Data Plane Development Kit (DPDK). Porém, você pode obter aceleração de pacotes conectando-se diretamente aos adaptadores Elastic Network Adapters (ENA) do Amazon EC2 por meio plugins de dispositivo host gerenciado e de `ipvlan` do Multus.
- O Amazon EKS suporta o Multus, que fornece um processo genérico que permite o encadeamento simples de plugins de CNI adicionais. Há suporte ao Multus e ao processo de encadeamento, mas a AWS não fornece suporte a todos os plugins de CNI compatíveis que podem ser encadeados ou para problemas que possam surgir nesses plugins de CNI que não estejam relacionados à configuração do encadeamento.
- O Amazon EKS está fornecendo suporte e gerenciamento do ciclo de vida para o plugin Multus, mas não é responsável por nenhum endereço IP ou gerenciamento adicional associado às interfaces de rede adicionais. O endereço IP e o gerenciamento da interface de rede padrão que utiliza o plugin Amazon VPC CNI permanecem inalterados.
- Somente o plugin Amazon VPC CNI recebe suporte oficial como o plugin delegado padrão. Você precisa modificar o manifesto de instalação publicado do Multus para reconfigurar o plugin

delegado padrão para uma CNI alternativa se você optar por não usar o plugin Amazon VPC CNI para redes primárias.

- O suporte ao Multus só é oferecido quando a Amazon VPC CNI é usada como a CNI principal. Não oferecemos suporte à Amazon VPC CNI quando usada para interfaces de ordem superior, secundárias ou não.
- Para evitar que o plugin Amazon VPC CNI tente gerenciar interfaces de rede adicionais atribuídas aos Pods, adicione a tag a seguir à interface de rede:

  chave: `node.k8s.amazonaws.com/no_manage`

  valor: `true`
- O Multus é compatível com políticas de rede, mas a política precisa ser enriquecida para incluir portas e endereços IP que talvez façam parte de interfaces de rede adicionais conectadas aos Pods.

Para uma demonstração de implementação, consulte o Multus Setup Guide (Guia de configuração do Multus) no GitHub.

Plugins CNI compatíveis alternativos

O Amazon VPC CNI plugin for Kubernetes é o único plug-in de CNI compatível com o Amazon EKS. O Amazon EKS é executado upstream no Kubernetes, para que você possa instalar plug-ins alternativos de CNI em nós do Amazon EC2 em seu cluster. Se você tiver nós do Fargate em seu cluster, o Amazon VPC CNI plugin for Kubernetes já estará em seus nós do Fargate. Trata-se do único plug-in de CNI que você pode usar com os nós do Fargate. Uma tentativa de instalar um plug-in alternativo de CNI em nós do Fargate falhará.

Se você quiser usar um plugin alternativo de CNI em nós do Amazon EC2, recomendamos que obtenha suporte comercial para o plug-in ou tenha a experiência interna para solucionar problemas e contribuir com correções para o projeto do plug-in de CNI.

O Amazon EKS mantém um relacionamento com uma rede de parceiros que oferecem suporte para plugins CNI compatíveis alternativos. Para obter mais detalhes sobre as versões, as qualificações e os testes realizados, consulte a documentação dos parceiros a seguir.

Parceiro	Produto	Documentação
Tigera	Calico	Instruções de instalação

Parceiro	Produto	Documentação
Isovalent	Cilium	Instruções de instalação
Juniper	Cloud-Native Contrail Networking (CN2)	Instruções de instalação
VMware	Antrea	Instruções de instalação

O Amazon EKS tem como objetivo oferecer uma ampla variedade de opções para cobrir todos os casos de uso.

Plugins de política de rede compatíveis alternativos

O Calico é uma solução amplamente adotada para rede e segurança de contêineres. Usar o Calico no EKS fornece uma aplicação de política de rede totalmente compatível para seus clusters do EKS. Além disso, você pode optar por usar a rede do Calico, que conserva os endereços IP da sua VPC subjacente. O Calico Cloud aprimora os recursos do Calico Open Source, fornecendo recursos avançados de segurança e observabilidade.

O que é o AWS Load Balancer Controller?

O AWS Load Balancer Controller gerencia os AWS Elastic Load Balancers para um cluster do Kubernetes. É possível usar o controlador para expor as aplicações no cluster para a Internet. O controlador provisiona balanceadores de carga da AWS que realizam o direcionamento para os recursos Service ou Ingress do cluster. Em outras palavras, o controlador cria um único endereço IP ou nome DNS que realiza o direcionamento para múltiplos pods em seu cluster.

O controlador monitora os recursos Ingress ou Service do Kubernetes. Em resposta, ele cria os recursos apropriados do AWS Elastic Load Balancing. É possível configurar um comportamento específico para os balanceadores de carga ao aplicar anotações aos recursos do Kubernetes. Por exemplo, você pode anexar grupos de segurança da AWS a balanceadores de carga ao usar anotações.

O controlador fornece os seguintes recursos:

Kubernetes `Ingress`

O LBC cria um AWS Application Load Balancer (ALB) quando você cria um recurso `Ingress` do Kubernetes. Faça uma análise das anotações que você pode aplicar a um recurso Ingress.

Serviço Kubernetes do tipo `LoadBalancer`

O LBC cria um AWS Network Load Balancer (NLB) quando você cria um serviço do Kubernetes do tipo `LoadBalancer`. Faça uma análise das anotações que você pode aplicar a um recurso Service.

Antigamente, o Network Load Balancer do Kubernetes era usado para destinos de instância, mas usava o LBC para destinos de IP. Com o AWS Load Balancer Controller versão `2.3.0` ou posterior, você pode criar NLBs usando qualquer tipo de destino. Para obter mais informações sobre os tipos de destino do NLB consulte Tipo de destino no Guia do usuário do Network Load Balancer.

O controlador é um projeto de código aberto gerenciado no GitHub.

Antes de implantar o controlador, recomendamos que você analise os pré-requisitos e as considerações em Aplicação de roteamento e tráfego HTTP com Application Load Balancers e Roteamento TCP e tráfego UDP com Network Load Balancers. Nesses tópicos, você implantará uma aplicação de amostra que inclui um balanceador de carga da AWS.

Instalar o controlador

- Saiba como the section called “Instalação usando o Helm”. Use este procedimento se você não tiver familiaridade com o Amazon EKS. Este procedimento usa o Helm, um gerenciador de pacotes para o Kubernetes, e a ferramenta `eksctl` para simplificar a instalação do LBC.
- Como alternativa, the section called “Instalação com manifestos”. Este procedimento é apropriado para configurações avançadas de cluster. Isso inclui clusters com acesso restrito à rede para registros de contêineres públicos.

Migrar de versões descontinuadas do controlador

- Se você tiver versões descontinuadas do AWS Load Balancer Controller instaladas, aprenda como the section called “Migração do controlador descontinuado”.
- Não é possível atualizar versões descontinuadas. Essas versões devem ser removidas e uma versão atual do AWS Load Balancer Controller deve ser instalada.
- As versões descontinuadas incluem:
  - AWS ALB Ingress Controller para Kubernetes (“Ingress Controller”), um antecessor do AWS Load Balancer Controller.

- Qualquer versão `0.1.x` do AWS Load Balancer Controller.

Provedor de nuvem herdado

O Kubernetes inclui um provedor de nuvem herdado para a AWS. O provedor de nuvem herdado é capaz de provisionar balanceadores de carga da AWS, que são semelhantes ao AWS Load Balancer Controller. O provedor de nuvem herdado cria Classic Load Balancers. Se você não instalar o AWS Load Balancer Controller, o Kubernetes usará o provedor de nuvem herdado como padrão. Você deve instalar o AWS Load Balancer Controller e evitar o uso do provedor de nuvem herdado.

> **Important**
>
> Nas versões 2.5 e mais recentes, o AWS Load Balancer Controller passa a ser o controlador padrão para os recursos do tipo serviço do Kubernetes com o `type: LoadBalancer` e faz um Network Load Balancer (NLB) da AWS para cada serviço. Ele faz isso criando um webhook mutante para serviços, que define o campo `spec.loadBalancerClass` como `service.k8s.aws/nlb` para novos serviços `type: LoadBalancer`. Você pode desativar esse recurso e voltar a usar o provedor de nuvem antigo como o controlador padrão, definindo o valor `enableServiceMutatorWebhook` do chart do Helm para `false`. A menos que você desative esse recurso, o cluster não provisionará novos Classic Load Balancers para seus serviços. Os Classic Load Balancers existentes continuarão funcionando.

Instalação do AWS Load Balancer Controller usando o Helm

Este tópico descreve como instalar o AWS Load Balancer Controller usando o Helm, um gerenciador de pacotes para o Kubernetes, e a ferramenta `eksctl`. O controlador é instalado com as opções padrão. Para obter mais informações sobre o controlador, incluindo detalhes sobre como configurá-lo com anotações, consulte a documentação do AWS Load Balancer Controller no GitHub.

Nas etapas a seguir, substitua *`example values`* pelos seus próprios valores:

Pré-requisitos

Antes de iniciar este tutorial, você deve instalar e configurar as ferramentas a seguir e os recursos necessários para criar e gerenciar um cluster do Amazon EKS.

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS.

- Um provedor OpenID Connect (OIDC) do AWS Identity and Access Management (IAM) existente para o cluster. Para determinar se você tem ou para criar uma, consulte Criar um provedor OIDC do IAM para o cluster.
- Certifique-se de que os complementos Amazon VPC CNI plugin for Kubernetes, `kube-proxy` e CoreDNS sejam das versões mínimas listadas em Tokens de contas de serviços.
- Familiaridade com o AWS Elastic Load Balancing. Para obter mais informações, consulte o Manual do usuário do Elastic Load Balancing.
- Familiaridade com o serviço e os recursos de ingresso do Kubernetes.

- Instalação do Helm localmente.

Etapa 1: criar um perfil do IAM usando a ferramenta **`eksctl`**

> **Note**
>
> Você precisa criar somente um perfil do IAM para o AWS Load Balancer Controller por conta da AWS. Verifique se `AmazonEKSLoadBalancerControllerRole` existe no Console do IAM. Se esse perfil existir, prossiga para the section called “Etapa 2: instalar o AWS Load Balancer Controller”.

Crie uma política do IAM.

1. Baixe uma política do IAM para o AWS Load Balancer Controller que permita que ele faça chamadas para APIs da AWS em seu nome.

   AWS

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy.json
   ```

   AWS GovCloud (US)

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy_us-gov.json
   ```

```
$ mv iam_policy_us-gov.json iam_policy.json
```

2. Crie uma política do IAM usando a política obtida por download na etapa anterior.

```
$ aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

> **Note**
>
> Se você visualizar a política no AWS Management Console, ele mostrará os avisos para o serviço ELB, mas não para o serviço ELB v2. Isso acontece porque algumas das ações da política existem para o ELB v2, mas não para o ELB. Você pode ignorar esses avisos para o ELB..

Criar um perfil do IAM usando a ferramenta **`eksctl`**

- Substitua *`my-cluster`* pelo nome do seu cluster e *`111122223333`* pelo ID da conta e, depois, execute o comando. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
$ eksctl create iamserviceaccount \
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-
arn=arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve
```

Etapa 2: instalar o AWS Load Balancer Controller

Instalar o AWS Load Balancer Controller usando o Helm V3

1. Adicione o repositório de chart do Helm `eks-charts`. A AWS mantém esse repositório no GitHub.

```
$ helm repo add eks https://aws.github.io/eks-charts
```

2. Atualize o repositório local para confirmar que você tem os gráficos mais recentes.

```
$ helm repo update eks
```

3. Instale o AWS Load Balancer Controller.

   Substitua o *my-cluster* pelo nome do cluster. No comando a seguir, `aws-load-balancer-controller` é a conta de serviço do Kubernetes que você criou em uma etapa anterior.

   Para obter mais informações sobre como configurar o chart do Helm, consulte `values.yaml` no GitHub.

```
$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=my-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

   a. Se você estiver implantando o controlador nos nós do Amazon EC2 aos quais você tem acesso restrito ao Instance Metadata Service (IMDS – Serviço de metadados de instância) do Amazon EC2 ou se você estiver implantando no Fargate, adicione os seguintes sinalizadores ao comando `helm`:

      - **--set region=*region-code***
      - **--set vpcId=*vpc-xxxxxxxx***

   b. Para visualizar as versões disponíveis do Chart do Helm e do Load Balancer Controller, use o seguinte comando:

```
helm search repo eks/aws-load-balancer-controller --versions
```

> **Important**
>
> O gráfico implantado não recebe atualizações de segurança automaticamente. Você precisa atualizar manualmente para um gráfico mais recente quando ele estiver

disponível. Ao realizar uma atualização, altere *`install`* para **`upgrade`** no comando anterior.
O comando `helm install` instala automaticamente as definições de recursos personalizados (CRDs) para o controlador. O comando `helm upgrade` não realiza essa instalação. Caso use `helm upgrade`,, você deverá instalar manualmente as CRDs. Execute o seguinte comando para instalar as CRDs:

```
wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-controller/crds/crds.yaml
kubectl apply -f crds.yaml
```

Etapa 3: verificar se o controlador está instalado

1. Verifique se o controlador está instalado.

   ```
   $ kubectl get deployment -n kube-system aws-load-balancer-controller
   ```

   Veja um exemplo de saída abaixo.

   ```
   NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
   aws-load-balancer-controller   2/2     2            2           84s
   ```

   Você recebe a saída anterior se tiver implantado usando o Helm. Se você implantou usando o manifesto do Kubernetes, só tem uma réplica.
2. Antes de usar o controlador para provisionar os recursos da AWS, o cluster deverá cumprir requisitos específicos. Para ter mais informações, consulte Aplicação de roteamento e tráfego HTTP com Application Load Balancers e Roteamento TCP e tráfego UDP com Network Load Balancers.

Instalação do complemento AWS Load Balancer Controller usando manifestos do Kubernetes

Este tópico descreve como instalar o controlador ao fazer download e aplicar os manifestos do Kubernetes. Você pode visualizar a documentação completa para o controlador no GitHub.

Nas etapas a seguir, substitua *`example values`* pelos seus próprios valores:

Pré-requisitos

Antes de iniciar este tutorial, você deve instalar e configurar as ferramentas a seguir e os recursos necessários para criar e gerenciar um cluster do Amazon EKS.

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS.
- Um provedor OpenID Connect (OIDC) do AWS Identity and Access Management (IAM) existente para o cluster. Para determinar se você tem ou para criar uma, consulte Criar um provedor OIDC do IAM para o cluster.
- Certifique-se de que os complementos Amazon VPC CNI plugin for Kubernetes, `kube-proxy` e CoreDNS sejam das versões mínimas listadas em Tokens de contas de serviços.
- Familiaridade com o AWS Elastic Load Balancing. Para obter mais informações, consulte o Manual do usuário do Elastic Load Balancing.
- Familiaridade com o serviço e os recursos de ingresso do Kubernetes.

Etapa 1: configurar o IAM

> **Note**
>
> Você precisa criar somente um perfil do IAM para o AWS Load Balancer Controller por conta da AWS. Verifique se `AmazonEKSLoadBalancerControllerRole` existe no Console do IAM. Se esse perfil existir, prossiga para the section called “Etapa 2: instalar o `cert-manager`”.

Crie uma política do IAM.

1. Baixe uma política do IAM para o AWS Load Balancer Controller que permita que ele faça chamadas para APIs da AWS em seu nome.

   AWS

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy.json
   ```

AWS GovCloud (US)

```
$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy_us-gov.json
```

```
$ mv iam_policy_us-gov.json iam_policy.json
```

2. Crie uma política do IAM usando a política obtida por download na etapa anterior.

```
$ aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

> **Note**
>
> Se você visualizar a política no AWS Management Console, ele mostrará os avisos para o serviço ELB, mas não para o serviço ELB v2. Isso acontece porque algumas das ações da política existem para o ELB v2, mas não para o ELB. Você pode ignorar esses avisos para o ELB..

eksctl

Criar um perfil do IAM usando a ferramenta **`eksctl`**

- Substitua *`my-cluster`* pelo nome do seu cluster e *`111122223333`* pelo ID da conta e, depois, execute o comando. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
$ eksctl create iamserviceaccount \
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-
arn=arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve
```

AWS CLI and kubectl

Criar um perfil do IAM usando a AWS CLI e a ferramenta **kubectl**

1. Recupere o ID do provedor do OIDC do cluster e armazene-a em uma variável.

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

2. Determine se um provedor do OIDC do IAM com seu ID de cluster já está em sua conta. Você precisa do OIDC configurado para o cluster e para o IAM.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

   Se um resultado for retornado, significará que você já tem um provedor OIDC do IAM para o cluster. Se nenhum resultado for retornado, você deverá criar um provedor OIDC do IAM para seu cluster. Para ter mais informações, consulte Criar um provedor OIDC do IAM para o cluster.

3. Copie o conteúdo a seguir para o seu dispositivo. Substitua *111122223333* pelo ID da sua conta. Substitua *region-code* pela Região da AWS em que está o cluster. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* pela saída retornada na etapa anterior. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`. Após substituir o texto, execute o comando modificado para criar o arquivo `load-balancer-role-trust-policy.json`.

```
cat >load-balancer-role-trust-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
```

```
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:aws-load-balancer-controller"
                }
            }
        }
    ]
}
EOF
```

4. Crie o perfil do IAM.

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

5. Anexe a política de IAM gerenciada pelo Amazon EKS ao perfil do IAM. Substitua *111122223333* pelo ID da sua conta.

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
  --role-name AmazonEKSLoadBalancerControllerRole
```

6. Copie o conteúdo a seguir para o seu dispositivo. Substitua *111122223333* pelo ID da sua conta. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`. Após substituir o texto, execute o comando modificado para criar o arquivo `aws-load-balancer-controller-service-account.yaml`.

```
cat >aws-load-balancer-controller-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn:
 arn:aws:iam::111122223333:role/AmazonEKSLoadBalancerControllerRole
```

```
EOF
```

7. Crie a conta de serviço Kubernetes no cluster: A conta de serviço do Kubernetes denominada `aws-load-balancer-controller` é anotado com o perfil do IAM criado com o nome *`AmazonEKSLoadBalancerControllerRole`*.

```
$ kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

Etapa 2: instalar o **`cert-manager`**

Instale `cert-manager` usando um dos métodos a seguir para injetar a configuração do certificado nos webhooks. Para obter mais informações, consulte Introdução na *`cert-manager`*Documentação.

Recomendamos usar o registro de contêiner `quay.io` para instalar `cert-manager`. Se os nós não tiverem acesso ao registro de contêiner `quay.io`, instale `cert-manager` usando o Amazon ECR (veja abaixo).

Quay.io

Instalar **`cert-manager`** usando Quay.io

- Se os nós tiverem acesso ao registro do contêiner `quay.io`, instale `cert-manager` para injetar a configuração do certificado nos webhooks.

```
$ kubectl apply \
    --validate=false \
    -f https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-
manager.yaml
```

Amazon ECR

Instalar **`cert-manager`** usando o Amazon ECR

1. Instale `cert-manager` usando um dos métodos a seguir para injetar a configuração do certificado nos webhooks. Para obter mais informações, consulte Introdução na *`cert-manager`*Documentação.
2. Faça download do manifesto.

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/
download/v1.13.5/cert-manager.yaml
```

3. Extraia as imagens a seguir e envie-as para um repositório ao qual seus nós têm acesso. Para obter mais informações sobre como extrair, etiquetar e enviar as imagens para seu próprio repositório, consulte Copiar uma imagem de contêiner de um repositório para outro.

```
quay.io/jetstack/cert-manager-cainjector:v1.13.5
quay.io/jetstack/cert-manager-controller:v1.13.5
quay.io/jetstack/cert-manager-webhook:v1.13.5
```

4. Substitua `quay.io` no manifesto nas três imagens pelo seu próprio nome de registro. O comando a seguir pressupõe que o nome do repositório privado seja o mesmo que o repositório de origem. Substitua *`111122223333.dkr.ecr.region-code.amazonaws.com`* pelo seu registro privado.

```
$ sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./
cert-manager.yaml
```

5. Aplique o manifesto.

```
$ kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

Etapa 3: instalar o AWS Load Balancer Controller

Instalar o AWS Load Balancer Controller usando um manifesto do Kubernetes

1. Faça download da especificação do controlador. Para obter mais informações sobre o controlador, consulte a documentação do GitHub.

```
curl -Lo v2_7_2_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-
controller/releases/download/v2.7.2/v2_7_2_full.yaml
```

2. Faça as edições a seguir ao arquivo.

   a. Se você tiver baixado o arquivo `v2_7_2_full.yaml`, execute o seguinte comando para remover a seção `ServiceAccount` no manifesto. Se você não remover essa seção,

a anotação necessária que você fez na conta de serviço em uma etapa anterior será substituída. A remoção dessa seção também preservará a conta de serviço criada em uma etapa anterior se você excluir o controlador.

```
$ sed -i.bak -e '612,620d' ./v2_7_2_full.yaml
```

Se você tiver baixado uma versão de arquivo diferente, abra o arquivo em um editor e remova as linhas a seguir.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
---
```

b. Substitua `your-cluster-name` na seção `Deployment spec` do arquivo pelo nome do cluster substituindo *`my-cluster`* pelo nome do seu cluster.

```
$ sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_7_2_full.yaml
```

c. Se seus nós não tiverem acesso aos repositórios de imagens do Amazon ECR do Amazon EKS, você precisará extrair a image a seguir a seguir e enviá-la a um repositório ao qual seus nós tenham acesso. Para obter mais informações sobre como extrair, etiquetar e enviar uma imagem para seu próprio repositório, consulte Copiar uma imagem de contêiner de um repositório para outro.

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.7.2
```

Adicione o nome do registro ao manifesto. O comando a seguir pressupõe que o nome do repositório privado seja o mesmo que o repositório de origem e adicione o nome de seu registro privado para o arquivo. Substitua *`111122223333.dkr.ecr.region-code.amazonaws.com`* pela seu registro. Essa linha pressupõe que você nomeou seu repositório privado da mesma forma que o repositório de origem. Caso contrário, altere o texto de `eks/aws-load-balancer-controller` após o nome do registro privado para o nome do repositório.

```
$ sed -i.bak -e 's|public.ecr.aws/eks/aws-load-balancer-
controller|111122223333.dkr.ecr.region-code.amazonaws.com/eks/aws-load-
balancer-controller|' ./v2_7_2_full.yaml
```

d. (Obrigatório somente para o Fargate ou para um IMDS restrito)

Se você estiver implantando o controlador nos nós do Amazon EC2 aos quais você tem acesso restrito ao serviço de metadados de instância (IMDS - instance metadata service) do Amazon EC2 ou se você estiver implantando no Fargate, adicione os **`following parameters`** em `- args:`.

```
[...]
spec:
      containers:
        - args:
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code


[...]
```

3. Aplique o arquivo.

```
$ kubectl apply -f v2_7_2_full.yaml
```

4. Baixe os manifestos `IngressClass` e `IngressClassParams` para seu cluster.

```
$ curl -Lo v2_7_2_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-
balancer-controller/releases/download/v2.7.2/v2_7_2_ingclass.yaml
```

5. Aplique o manifesto ao cluster.

```
$ kubectl apply -f v2_7_2_ingclass.yaml
```

Etapa 4: verificar se o controlador está instalado

1. Verifique se o controlador está instalado.

```
$ kubectl get deployment -n kube-system aws-load-balancer-controller
```

Veja um exemplo de saída abaixo.

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

Você recebe a saída anterior se tiver implantado usando o Helm. Se você implantou usando o manifesto do Kubernetes, só tem uma réplica.

2. Antes de usar o controlador para provisionar os recursos da AWS, o cluster deverá cumprir requisitos específicos. Para ter mais informações, consulte Aplicação de roteamento e tráfego HTTP com Application Load Balancers e Roteamento TCP e tráfego UDP com Network Load Balancers.

Migração do controlador descontinuado

Este tópico descreve como migrar de versões descontinuadas do controlador. Mais especificamente, ele descreve como remover versões descontinuadas do AWS Load Balancer Controller.

- Não é possível atualizar versões descontinuadas. As versões descontinuadas devem ser removidas e uma versão atual do LBC deve ser instalada.
- As versões descontinuadas incluem:
  - AWS ALB Ingress Controller para Kubernetes (“Ingress Controller”), um antecessor do AWS Load Balancer Controller.
  - Qualquer versão `0.1.`*x* do AWS Load Balancer Controller.

Remoção da versão descontinuada do controlador

> **Note**
>
> É possível que você tenha instalado a versão descontinuada usando o Helm ou de forma manual com manifestos do Kubernetes. Conclua o procedimento utilizando a ferramenta com a qual ele foi originalmente instalado.

Remover o Ingress Controller usando o Helm

1. Se você instalou o chart do Helm `incubator/aws-alb-ingress-controller`, desinstale-o.

```
$ helm delete aws-alb-ingress-controller -n kube-system
```

2. Se tiver a versão `0.1.x` do chart `eks-charts/aws-load-balancer-controller` instalado, desinstale-a. O upgrade de `0.1.x` para a versão `1.0.0` não funciona por causa de uma incompatibilidade com a versão da API do webhook.

```
$ helm delete aws-load-balancer-controller -n kube-system
```

Remover o Ingress Controller usando o manifesto do Kubernetes

1. Verifique se o controlador está instalado no momento.

```
$ kubectl get deployment -n kube-system alb-ingress-controller
```

   Esta é a saída se o controlador não estiver instalado.

   Error from server (NotFound): deployments.apps "alb-ingress-controller" not found

   Esta é a saída se o controlador estiver instalado.

```
NAME                   READY UP-TO-DATE AVAILABLE AGE
alb-ingress-controller 1/1   1          1         122d
```

2. Insira o comando a seguir para executar o controlador.

```
$ kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-
ingress-controller/v1.1.8/docs/examples/alb-ingress-controller.yaml
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-
ingress-controller/v1.1.8/docs/examples/rbac-role.yaml
```

Migração para o AWS Load Balancer Controller

Para migrar do ALB Ingress Controller para Kubernetes para o AWS Load Balancer Controller, você precisa:

1. Remover o ALB Ingress Controller (veja as etapas anteriores).
2. Instalar o AWS Load Balancer Controller.
3. Adicionar uma política adicional para o perfil do IAM usado pelo LBC. Essa política permite que o LBC gerencie recursos criados pelo ALB Ingress Controller para Kubernetes.

Adicionar a política de migração ao perfil do IAM do AWS Load Balancer Controller.

1. Faça download da política do IAM. Essa política permite que o LBC gerencie recursos criados pelo ALB Ingress Controller para Kubernetes. Você também pode visualizar a política .

```
$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.7.2/docs/install/iam_policy_v1_to_v2_additional.json
```

2. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`..

```
$ sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

3. Crie a política do IAM e observe o ARN retornado.

```
$ aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

4. Anexe a política do IAM ao perfil do IAM usado pelo LBC. Substitua *`your-role-name`* pelo nome do perfil, por exemplo, `AmazonEKSLoadBalancerControllerRole`.

   Se você tiver criado o perfil usando `eksctl`, para localizar o nome do perfil que foi criado, abra o console do AWS CloudFormation e selecione a pilha eksctl-***`my-cluster`***-addon-iamserviceaccount-kube-system-aws-load-balancer-controller. Selecione a guia Recursos. O nome da função está na coluna ID físico. Se o cluster estiver nas Regiões da AWS: AWS GovCloud (EUA-Leste) ou AWS GovCloud (EUA-Oeste), substitua `arn:aws:` por `arn:aws-us-gov:`.

```
$ aws iam attach-role-policy \
  --role-name your-role-name \
  --policy-arn
 arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerAdditionalIAMPolicy
```

Trabalhando com o complemento CoreDNS do Amazon EKS

O CoreDNS é um servidor DNS flexível e extensível que pode servir como DNS de cluster do Kubernetes. Quando você executa um cluster do Amazon EKS com pelo menos um nó, duas réplicas da imagem do CoreDNS são implantadas por padrão, independentemente do número de nós implantados em seu cluster. Os CoreDNS Pods fornecem resolução de nomes para todos os Pods no cluster. Os Pods do CoreDNS poderão ser implantados em nós do Fargate se o seu cluster incluir Defina quais Pods usarão o AWS Fargate quando em execução com um namespace correspondente ao namespace para a `deployment` do CoreDNS. Para obter mais informações sobre o CoreDNS, consulte Uso do CoreDNS para descoberta de serviço na documentação do Kubernetes.

A tabela a seguir lista a versão mais recente do tipo de complemento do Amazon EKS para cada versão do Kubernetes.

Versão do Kubernetes	**1.30**	**1.29**	**1.28**	**1.27**	**1.26**	**1.25**	**1.24**	**1.23**
	v1.11.1-eksbuild	v1.11.1-eksbuild	v1.10.1-eksbuild.1	v1.10.1-eksbuild.1	v1.9.3-eksbuild.	v1.9.3-eksbuild.	v1.9.3-eksbuild.	v1.8.7-eksbuild.10

> **Important**
>
> Se você estiver gerenciando esse complemento automaticamente, as versões na tabela podem não ser as mesmas que as versões autogerenciadas disponíveis. Para obter mais informações sobre como atualizar o tipo autogerenciado desse complemento, consulte Atualização do complemento autogerenciado.

Considerações importantes sobre o upgrade do CoreDNS

- Para melhorar a estabilidade e a disponibilidade do CoreDNS Deployment, as versões `v1.9.3-eksbuild.6` e posteriores e o `v1.10.1-eksbuild.3` são implantados com um `PodDisruptionBudget`. Se você implantou um `PodDisruptionBudget` existente, sua atualização para essas versões pode falhar. Se a atualização falhar, concluir uma das seguintes tarefas deve resolver o problema:

  - Ao fazer a atualização do complemento do Amazon EKS, opte por substituir as configurações existentes como sua opção de resolução de conflitos. Se você fez outras configurações personalizadas no Deployment, certifique-se de fazer backup de suas configurações antes de atualizar para poder reaplicar as demais configurações personalizadas após a atualização.
  - Remova o `PodDisruptionBudget` existente e tente fazer a atualização novamente.
- Nas versões complementares do EKS `v1.9.3-eksbuild.3` e posteriores `v1.10.1-eksbuild.6` e posteriores, ele CoreDNS Deployment define o `readinessProbe` para usar o `/ready` endpoint. Esse endpoint está habilitado no `Corefile` arquivo de configuração do CoreDNS.

  Se você usar um personalizado`Corefile`, deverá adicionar o `ready` plug-in à configuração, para que o `/ready` endpoint fique ativo CoreDNS para o teste ser usado.
- Nas versões complementares do EKS `v1.9.3-eksbuild.7` e posteriores e `v1.10.1-eksbuild.4` e posteriores, você pode alterar o `PodDisruptionBudget`. Você pode editar o complemento e alterar essas configurações nas configurações opcionais usando os campos no exemplo a seguir. Este exemplo mostra o padrão `PodDisruptionBudget`.

```
{
    "podDisruptionBudget": {
        "enabled": true,
        "maxUnavailable": 1
        }
}
```

  Você pode definir `maxUnavailable` ou `minAvailable`, mas não pode definir os dois em um único `PodDisruptionBudget`. Para obter mais informações sobre `PodDisruptionBudgets`, consulte ReplicaSetPodDisruptionBudget na Kubernetes documentação.

  Observe que se você definir `enabled` como `false`, o `PodDisruptionBudget` não será removido. Depois de definir esse campo como `false`, você deve excluir o `PodDisruptionBudget` objeto. Da mesma forma, se você editar o complemento para usar uma versão mais antiga do complemento (rebaixar o complemento) depois de atualizar para uma versão com um `PodDisruptionBudget`, o não será removido `PodDisruptionBudget`. Para excluir o `PodDisruptionBudget`, você pode executar o comando a seguir:

```
kubectl delete poddisruptionbudget coredns -n kube-system
```

- No complemento do EKS versão `v1.10.1-eksbuild.5` e posteriores, altere a tolerância padrão de `node-role.kubernetes.io/master:NoSchedule` para `node-role.kubernetes.io/control-plane:NoSchedule` para manter a conformidade com o KEP 2067. Para obter mais informações sobre o KEP 2067, consulte KEP-2067: renomear o rótulo "master" e taint kubeadm nas Propostas de aprimoramento do Kubernetes (KEPs) no GitHub.

  No complemento do EKS versão `v1.8.7-eksbuild.8` e posteriores e `v1.9.3-eksbuild.9` e posteriores, ambas as tolerâncias são configuradas para serem compatíveis com todas as versões do Kubernetes.
- No complemento do EKS versão `v1.9.3-eksbuild.11` e `v1.10.1-eksbuild.7` e posteriores, o CoreDNS Deployment define um valor padrão para `topologySpreadConstraints`. O valor padrão garante que os Pods do CoreDNS estejam espalhados pelas zonas de disponibilidade se houver nós em várias zonas de disponibilidade disponíveis. Você pode definir um valor personalizado que será usado em vez do valor padrão. O valor padrão é:

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

Considerações sobre upgrade do CoreDNS v**1.11**

- No complemento do EKS versão `v1.11.1-eksbuild.4` e posteriores, a imagem de contêiner é baseada em uma imagem de base mínima mantida pelo Amazon EKS Distro, que contém pacotes mínimos e não tem shells. Para obter mais informações, consulte Amazon EKS Distro. O uso e a solução de problemas da imagem do CoreDNS permanecem os mesmos.

Criar o complemento do Amazon EKS

Crie o tipo do Amazon EKS do complemento. Verificar

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS.

1. Veja qual versão do complemento está atualmente instalada no cluster.

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -d : -f 3
```

Veja um exemplo de saída abaixo.

```
v1.10.1-eksbuild.11
```

2. Veja qual tipo de complemento está atualmente instalado no cluster. Dependendo da ferramenta com a qual você criou o cluster, talvez você não tenha o tipo de complemento do Amazon EKS instalado em seu cluster atualmente. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query addon.addonVersion --output text
```

Se um número de versão for retornado, você tem o tipo de complemento do Amazon EKS instalado no cluster, e não precisa completar as etapas restantes deste procedimento. Se um erro for retornado, você não tem o tipo de complemento do Amazon EKS instalado no cluster. Conclua as etapas restantes desse procedimento para instalá-lo.

3. Salve a configuração do complemento instalado atualmente.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

4. Crie o complemento usando o AWS CLI. Se você quiser usar o AWS Management Console ou `eksctl` para criar o complemento, consulte Criar um complemento do Amazon EKS e especifique `coredns` como o nome do complemento. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.

- Substitua o *my-cluster* pelo nome do cluster.
- Substitua *v1.11.1-eksbuild.9* pela versão mais recente listada na tabela das versões mais recentes da versão do seu cluster.

```
aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-version v1.11.1-eksbuild.9
```

Se você aplicou configurações personalizadas ao seu complemento atual que entrem em conflito com as configurações padrão do complemento Amazon EKS, a criação poderá falhar. Se a criação falhar, você receberá um erro que poderá ajudar a resolver o problema. Como alternativa, você pode adicionar **`--resolve-conflicts OVERWRITE`** ao comando anterior. Isso permite que o complemento substitua todas as configurações personalizadas existentes. Depois de criar o complemento, você pode atualizá-lo com suas configurações personalizadas.

5. Confirme se a versão mais recente do complemento para a versão Kubernetes do seu cluster foi adicionada ao cluster. Substitua o *`my-cluster`* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

Pode levar alguns segundos para que a criação do complemento seja concluída.

Veja um exemplo de saída abaixo.

```
v1.11.1-eksbuild.9
```

6. Se você fez configurações personalizadas no complemento original, antes de criar o complemento Amazon EKS, use a configuração que você salvou em uma etapa anterior para atualizar o complemento Amazon EKS com as configurações personalizadas.

Atualizar o complemento do Amazon EKS

Atualize o tipo do Amazon EKS do complemento. Se você não adicionou o tipo Amazon EKS do complemento ao cluster, adicione-o ou consulte Atualização do complemento autogerenciado, em vez de concluir esse procedimento.

1. Veja qual versão do complemento está atualmente instalada no cluster. Substitua *`my-cluster`* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 "addon.addonVersion" --output text
```

Veja um exemplo de saída abaixo.

```
v1.10.1-eksbuild.11
```

Se a versão retornada for a mesma da versão do Kubernetes do cluster na tabela de versões mais recente, você já tem a versão mais recente instalada no cluster e não precisa concluir o restante desse procedimento. Se você receber um erro, em vez de um número de versão no resultado, você não tem o tipo Amazon EKS do complemento instalado no cluster. Você precisa criar o complemento antes de poder atualizá-lo com este procedimento.

2. Salve a configuração do complemento instalado atualmente.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

3. Atualize o complemento usando a AWS CLI. Se você quiser usar o AWS Management Console ou `eksctl` para atualizar o complemento, consulte Atualizar um complemento do Amazon EKS. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.

   - Substitua o *my-cluster* pelo nome do cluster.
   - Substitua *v1.11.1-eksbuild.9* pela versão mais recente listada na tabela das versões mais recentes da versão do seu cluster.
   - A opção *PRESERVE* de **--resolve-conflicts** mantém os valores de configuração existentes para o complemento. Se você definiu valores personalizados para as configurações do complemento e não usar essa opção, o Amazon EKS sobrescreverá seus valores pelos valores padrão. Se você usar essa opção, recomendamos testar qualquer alteração de campo e valor em um cluster que não seja de produção antes de atualizar o complemento no cluster de produção. Se você alterar esse valor para `OVERWRITE`, todas as configurações serão alteradas para os valores padrão do Amazon EKS. Se você definiu valores personalizados para qualquer configuração, eles poderão ser sobrescritos pelos valores padrão do Amazon EKS. Se você alterar esse valor para `none`, o Amazon EKS não alterará o valor de nenhuma configuração, mas a atualização poderá falhar. Se a atualização falhar, você receberá uma mensagem de erro para ajudar a resolver o conflito.
   - Se você não estiver atualizando uma configuração, remova **--configuration-values '{*"replicaCount":3*}'** do comando. Se você estiver atualizando uma definição de configuração, substitua *"replicaCount":3* pela definição que deseja definir. Neste exemplo, o número de réplicas de CoreDNS é definido como `3`. O valor que você especificar deve ser válido para o esquema da configuração. Se não souber qual é o esquema de configuração, execute **aws eks describe-addon-configuration --addon-name coredns --addon-version *v1.11.1-eksbuild.9***, substituindo *v1.11.1-eksbuild.9* pelo número da versão do complemento cuja configuração você deseja ver. O

esquema é retornado na saída. Se você tiver alguma configuração personalizada existente que deseja remover e definir os valores de todas as configurações de volta aos padrões do Amazon EKS, remova *"replicaCount":3* do comando para que você tenha arquivos vazios **{}**. Para obter mais informações sobre as configurações do CoreDNS, consulte Personalizando o Serviço DNS na documentação do Kubernetes.

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns --addon-
version v1.11.1-eksbuild.9 \
    --resolve-conflicts PRESERVE --configuration-values '{"replicaCount":3}'
```

Pode levar alguns segundos para que a atualização seja concluída.

4. Confirme se a versão do complemento foi atualizada. Substitua o *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

Pode levar alguns segundos para que a atualização seja concluída.

Veja um exemplo de saída abaixo.

```
{
    "addon": {
        "addonName": "coredns",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.11.1-eksbuild.9",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/coredns/
d2c34f06-1111-2222-1eb0-24f64ce37fa4",
        "createdAt": "2023-03-01T16:41:32.442000+00:00",
        "modifiedAt": "2023-03-01T18:16:54.332000+00:00",
        "tags": {},
        "configurationValues": "{\"replicaCount\":3}"
    }
}
```

Atualização do complemento autogerenciado

> **Important**
>
> Recomendamos adicionar o tipo Amazon EKS do complemento ao seu cluster em vez de usar o tipo autogerenciado do complemento. Se você não estiver familiarizado com a diferença entre os tipos, consulte the section called “Complementos do Amazon EKS”. Para obter mais informações sobre como adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criar um complemento”. Se você não conseguir usar o complemento do Amazon EKS, recomendamos que você envie um problema sobre o motivo pelo qual não pode usar o repositório GitHub para roteiro de contêineres.

1. Confirme que tem o tipo autogerenciado de complemento instalado em seu cluster. Substitua *my-cluster* pelo nome do cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se receber uma mensagem de erro, você tem o tipo autogerenciado do complemento instalado no cluster. Conclua as etapas restantes neste procedimento. Se receber um número de versão, você tem o tipo de complemento do Amazon EKS instalado no cluster. Para atualizar o tipo de complemento do Amazon EKS, use o procedimento em Atualizar o complemento do Amazon EKS, em vez de usar este procedimento. Se não estiver familiarizado com a diferença entre os tipos de complemento, consulte Complementos do Amazon EKS.

2. Veja qual versão da imagem do contêiner está atualmente instalada em seu cluster.

   ```
   kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
   ```

   Veja um exemplo de saída abaixo.

   ```
   v1.8.7-eksbuild.2
   ```

3. Se a versão atual do CoreDNS for `v1.5.0` ou superior, mas anterior à versão listada na tabela Versões do CoreDNS, ignore esta etapa. Se a sua versão atual for anterior à `1.5.0`, você precisa modificar o `ConfigMap` para que o CoreDNS use o complemento de encaminhamento, em vez do complemento de proxy.

1. Abra o arquivo configmap com o seguinte comando.

```
kubectl edit configmap coredns -n kube-system
```

2. Substitua `proxy` na linha a seguir pelo `forward`. Salve o arquivo e saia do editor.

```
proxy . /etc/resolv.conf
```

4. Se você implantou originalmente o cluster no Kubernetes `1.17` ou anterior, talvez seja necessário remover uma linha descontinuada do manifesto do CoreDNS.

> **Important**
>
> Você deve concluir essa etapa antes de atualizar para o CoreDNS versão `1.7.0`, mas é recomendável concluir essa etapa mesmo se estiver atualizando para uma versão anterior.

1. Verifique se o manifesto do CoreDNS tem a linha.

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
 grep upstream
```

Se nenhum resultado for retornado, o manifesto não terá a linha e você poderá prosseguir para a próxima etapa para atualizar o CoreDNS. Se um resultado for retornado, você precisará remover a linha.

2. Edite o `ConfigMap` com o comando a seguir, removendo a linha no arquivo que inclui a palavra `upstream` em seu nome. Não altere mais nada no arquivo. Depois que a linha for removida, salve as alterações.

```
kubectl edit configmap coredns -n kube-system -o yaml
```

5. Recupere a imagem atual do CoreDNS:

```
kubectl describe deployment coredns -n kube-system | grep Image
```

Veja um exemplo de saída abaixo.

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. Se você estiver atualizando para o CoreDNS `1.8.3` ou posterior, será necessário adicionar a permissão de `endpointslices` à `clusterrole` do `system:coredns` do Kubernetes.

```
kubectl edit clusterrole system:coredns -n kube-system
```

Adicione as linhas a seguir abaixo das linhas de permissões existentes na seção `rules` do arquivo.

```
[...]
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
[...]
```

7. Atualize o complemento CoreDNS substituindo *602401143452* e *region-code* pelos valores da saída retornados em uma etapa anterior. Substitua *v1.11.1-eksbuild.9* pela versão do CoreDNS listada na tabela de versões mais recentes para sua versão do Kubernetes.

```
kubectl set image deployment.apps/coredns -n kube-system
 coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.1-
eksbuild.9
```

Veja um exemplo de saída abaixo.

```
deployment.apps/coredns image updated
```

8. Verifique a versão da imagem do contêiner novamente para confirmar que ela foi atualizada para a versão que você especificou na etapa anterior.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

Veja um exemplo de saída abaixo.

```
v1.11.1-eksbuild.9
```

Autoescalabilidade do CoreDNS

Quando você executa um cluster do Amazon EKS com pelo menos um nó, uma Deployment de duas réplicas da imagem do CoreDNS é implantada por padrão, independentemente do número de nós implantados no seu cluster. Os pods do CoreDNS fornecem resolução de nomes para todos os pods no cluster. As aplicações usam a resolução de nomes para se conectar a pods e serviços no cluster, bem como para se conectar a serviços fora do cluster. À medida que o número de solicitações de resolução de nomes (consultas) dos pods aumenta, os pods do CoreDNS podem ficar sobrecarregados, se tornar lentos e rejeitar solicitações que não conseguem atender.

Para lidar com o aumento da carga nos pods do CoreDNS, considere um sistema de autoescalabilidade para o CoreDNS O Amazon EKS pode gerenciar a autoescalabilidade da implantação do CoreDNS na versão do complemeno do EKS do CoreDNS. Esse mecanismo de autoescalabilidade do CoreDNS monitora continuamente o estado do cluster, incluindo o número de nós e núcleos de CPU. Com base nessas informações, o controlador adaptará dinamicamente o número de réplicas da implantação do CoreDNS em um cluster do EKS. Esse recurso funciona para o CoreDNS `v1.9` e o release do EKS versão `1.25` e posteriores. Para obter mais informações sobre quais versões são compatíveis com a autoescalabilidade do CoreDNS, consulte a seção a seguir.

Recomendamos usar esse recurso em conjunto com outras práticas recomendadas de autoescalabilidade de clusters do EKS para melhorar a disponibilidade geral da aplicação e a escalabilidade do cluster.

Pré-requisitos

Para o Amazon EKS escalar sua implantação do CoreDNS, há três pré-requisitos:

- É necessário usar o complemento do EKS versão CoreDNS.
- Seu cluster deve executar pelo menos as versões mínimas do cluster e das versões da plataforma.
- Seu cluster deve executar pelo menos a versão mínima do complemento do EKS do CoreDNS.

Versão mínima do cluster

A autoescalabilidade do CoreDNS é feita por um novo componente no ambiente de gerenciamento do cluster, gerenciado pelo Amazon EKS. Por isso, você deve atualizar seu cluster para uma versão do EKS que ofereça suporte à versão mínima da plataforma que tem o novo componente.

Um novo cluster do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS. O cluster deve ser da versão `1.25` ou posterior do Kubernetes. O cluster deve executar uma das versões do Kubernetes e versões da plataforma listadas na tabela a seguir ou uma versão posterior. Observe que qualquer Kubernetes e da plataforma posteriores às versões listadas. Para verificar a versão atual do Kubernetes substitua *`my-cluster`* no comando a seguir pelo nome do cluster e execute o comando modificado: .

```
aws eks describe-cluster
              --name my-cluster --query cluster.version --output
              text
```

Versão do Kubernetes	Versão da plataforma
`1.29.3`	`eks.7`
`1.28.8`	`eks.13`
`1.27.12`	`eks.17`
`1.26.15`	`eks.18`
`1.25.16`	`eks.19`

> **Note**
>
> Todas as versões de plataforma ou versões posteriores do Kubernetes também são compatíveis, por exemplo, o Kubernetes versão `1.30` a partir de `eks.1` e em diante.

Versão mínima do complemento do EKS

Versão do Kubernetes	**1.29**	**1.28**	**1.27**	**1.26**	**1.25**
	`v1.11.1-eksbuild.9`	`v1.10.1-eksbuild.11`	`v1.10.1-eksbuild.11`	`v1.9.3-eksbuild.15`	`v1.9.3-eksbuild.15`

Configurar autoescalabilidade do CoreDNS no AWS Management Console

1. Certifique-se de que a versão do seu cluster seja igual ou superior à versão mínima do cluster.

   O Amazon EKS atualiza automaticamente clusters entre versões de plataforma da mesma versão do Kubernetes, e você não pode iniciar esse processo sozinho. Em vez disso, você pode atualizar seu cluster para a próxima versão do Kubernetes, e o cluster será atualizado para essa versão do K8s e a versão mais recente da plataforma. Por exemplo, se você fizer o upgrade de `1.25` para`1.26`, o cluster será atualizado para `1.26.15 eks.18`.

   As novas versões do Kubernetes às vezes introduzem alterações significativas. Portanto, recomendamos testar o comportamento das aplicações usando um cluster separado da nova versão do Kubernetes antes de atualizar os clusters de produção.

   Para atualizar um cluster para uma nova versão do Kubernetes, siga o procedimento em Atualizar um cluster existente para a nova versão do Kubernetes.
2. Certifique-se de ter o complemento do EKS para CoreDNS, e não a implantação do CoreDNS autogerenciada.

   Dependendo da ferramenta com a qual você criou o cluster, talvez você não tenha o tipo de complemento do Amazon EKS instalado em seu cluster atualmente. Para ver qual tipo de complemento está instalado no cluster, execute o comando a seguir. Substitua o `my-cluster` pelo nome do cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

Se um número de versão é devolvido, você tem o tipo de complemento do Amazon EKS instalado no cluster e pode prosseguir para a próxima etapa. Se um erro for retornado, você não tem o tipo de complemento do Amazon EKS instalado no cluster. Conclua as etapas restantes do procedimento Criar o complemento do Amazon EKS para substituir a versão autogerenciada pelo complemento do Amazon EKS.

3. Certifique-se de que seu complemento do EKS para CoreDNS seja de uma versão igual ou superior à versão mínima do complemento do EKS.

   Veja qual versão do complemento está atualmente instalada no cluster. É possível verificá-la no AWS Management Console ou executando o seguinte comando:

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -d : -f 3
   ```

   Veja um exemplo de saída abaixo.

   ```
   v1.10.1-eksbuild.11
   ```

   Compare esta versão à versão mínima do complemento do EKS na seção anterior. Se necessário, atualize o complemento do EKS para uma versão superior seguindo o procedimento Atualizar o complemento do Amazon EKS.

4. Adicione a configuração de autoescalabilidade às Configurações opcionais do complemento do EKS.

   a. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

   b. No painel de navegação esquerdo, selecione Clusters e depois o nome do cluster para o qual você deseja configurar o complemento.

   c. Escolha a guia Add-ons (Complementos).

   d. Selecione a caixa no canto superior direito da caixa do complemento do CoreDNS e, em seguida, escolha Editar.

   e. Na página Configurar CoreDNS:

      i. Selecione a Version (Versão) que você deseja usar. Recomendamos manter a mesma versão da etapa anterior e atualizar a versão e a configuração em ações separadas.

      ii. Expanda Definições de configuração opcionais.

iii. Insira a chave JSON `"autoscaling":` e o valor de um objeto JSON aninhado com uma chave `"enabled":` e o valor `true` em Valores de configuração. O texto resultante deve ser um objeto JSON válido. Se esse par de chave e valor for o único dado na caixa de texto, coloque-o entre colchetes `{}`. O seguinte exemplo mostra a autoescalabilidade habilitada:

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

iv. (Opcional) É possível fornecer valores mínimos e máximos para os quais a autoescalabilidade pode escalar o número de pods do CoreDNS.

O exemplo a seguir mostra que a autoescalabilidade está habilitada e todas as chaves opcionais têm valores. Recomendamos que o número mínimo de pods do CoreDNS seja sempre maior que 2 para fornecer resiliência ao serviço de DNS no cluster.

```
{
  "autoScaling": {
    "enabled": true,
    "minReplicas": 2,
    "maxReplicas": 10
  }
}
```

f. Para aplicar a nova configuração substituindo os pods do CoreDNS, escolha Salvar alterações.

O Amazon EKS aplica alterações nos complementos do EKS usando uma distribuição da implantação do Kubernetes para CoreDNS. É possível acompanhar o status da implantação no Histórico de atualizações do complemento no AWS Management Console e com `kubectl rollout status deployment/coredns --namespace kube-system`.

`kubectl rollout` oferece os seguintes comandos:

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
```

```
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Se a distribuição demorar muito, o Amazon EKS desfará a distribuição e uma mensagem com o tipo Atualização do complemento e o status Falha será adicionada ao Histórico de atualizações do complemento. Para investigar qualquer problema, comece com o histórico da distribuição e execute `kubectl logs` em um pod do CoreDNS para ver os logs do CoreDNS.

5. Se a nova entrada no Histórico de atualizações tiver o status de Êxito, a distribuição foi concluída e o complemento está usando a nova configuração em todos os pods do CoreDNS. Conforme você altera o número de nós e núcleos de CPU dos nós no cluster, o Amazon EKS escala o número de réplicas da implantação do CoreDNS.

Configurar autoescalabilidade do CoreDNS no AWS Command Line Interface

1. Certifique-se de que a versão do seu cluster seja igual ou superior à versão mínima do cluster.

   O Amazon EKS atualiza automaticamente clusters entre versões de plataforma da mesma versão do Kubernetes, e você não pode iniciar esse processo sozinho. Em vez disso, você pode atualizar seu cluster para a próxima versão do Kubernetes, e o cluster será atualizado para essa versão do K8s e a versão mais recente da plataforma. Por exemplo, se você fizer o upgrade de `1.25` para`1.26`, o cluster será atualizado para `1.26.15 eks.18`.

   As novas versões do Kubernetes às vezes introduzem alterações significativas. Portanto, recomendamos testar o comportamento das aplicações usando um cluster separado da nova versão do Kubernetes antes de atualizar os clusters de produção.

   Para atualizar um cluster para uma nova versão do Kubernetes, siga o procedimento em Atualizar um cluster existente para a nova versão do Kubernetes.

2. Certifique-se de ter o complemento do EKS para CoreDNS, e não a implantação do CoreDNS autogerenciada.

   Dependendo da ferramenta com a qual você criou o cluster, talvez você não tenha o tipo de complemento do Amazon EKS instalado em seu cluster atualmente. Para ver qual tipo de

complemento está instalado no cluster, execute o comando a seguir. Substitua o `my-cluster` pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

Se receber um número de versão, você tem o tipo de complemento do Amazon EKS instalado no cluster. Se um erro for retornado, você não tem o tipo de complemento do Amazon EKS instalado no cluster. Conclua as etapas restantes do procedimento Criar o complemento do Amazon EKS para substituir a versão autogerenciada pelo complemento do Amazon EKS.

3. Certifique-se de que seu complemento do EKS para CoreDNS seja de uma versão igual ou superior à versão mínima do complemento do EKS.

   Veja qual versão do complemento está atualmente instalada no cluster. É possível verificá-la no AWS Management Console ou executando o seguinte comando:

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   Veja um exemplo de saída abaixo.

   ```
   v1.10.1-eksbuild.11
   ```

   Compare esta versão à versão mínima do complemento do EKS na seção anterior. Se necessário, atualize o complemento do EKS para uma versão superior seguindo o procedimento Atualizar o complemento do Amazon EKS.

4. Adicione a configuração de autoescalabilidade às Configurações opcionais do complemento do EKS.

   Execute o seguinte comando AWS CLI. Substitua `my-cluster` pelo nome do cluster e o ARN do perfil do IAM pelo perfil que você está usando.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true}}'
   ```

O Amazon EKS aplica alterações nos complementos do EKS usando uma distribuição da implantação do Kubernetes para CoreDNS. É possível acompanhar o status da implantação no Histórico de atualizações do complemento no AWS Management Console e com `kubectl rollout status deployment/coredns --namespace kube-system`.

`kubectl rollout` oferece os seguintes comandos:

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Se a distribuição demorar muito, o Amazon EKS desfará a distribuição e uma mensagem com o tipo Atualização do complemento e o status Falha será adicionada ao Histórico de atualizações do complemento. Para investigar qualquer problema, comece com o histórico da distribuição e execute `kubectl logs` em um pod do CoreDNS para ver os logs do CoreDNS.

5. (Opcional) É possível fornecer valores mínimos e máximos para os quais a autoescalabilidade pode escalar o número de pods do CoreDNS.

   O exemplo a seguir mostra que a autoescalabilidade está habilitada e todas as chaves opcionais têm valores. Recomendamos que o número mínimo de pods do CoreDNS seja sempre maior que 2 para fornecer resiliência ao serviço de DNS no cluster.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true,"minReplicas":2,"maxReplicas":10}}'
   ```

6. Verifique o status da atualização do complemento executando o seguinte comando:

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns \
   ```

Se você vir esta linha: `"status": "ACTIVE"`, o lançamento foi concluído e o complemento está usando a nova configuração em todos os pods do CoreDNS. Conforme você altera o número de nós e núcleos de CPU dos nós no cluster, o Amazon EKS escala o número de réplicas da implantação do CoreDNS.

CoreDNS métricas

O CoreDNS como um complemento do EKS expõe as métricas do CoreDNS na porta `9153` no formato Prometheus no serviço `kube-dns`. É possível usar o Prometheus, o Amazon CloudWatch Agent ou qualquer outro sistema compatível para extrair (coletar) essas métricas.

Para ver um exemplo de configuração de extração compatível com o Prometheus e com o agente do CloudWatch, consulte Configuração do agente do CloudWatch para Prometheus no Guia do usuário do Amazon CloudWatch.

Trabalhando com o complemento **`kube-proxy`** do Kubernetes

> **Important**
>
> Recomendamos adicionar o tipo Amazon EKS do complemento ao seu cluster em vez de usar o tipo autogerenciado do complemento. Se você não estiver familiarizado com a diferença entre os tipos, consulte the section called “Complementos do Amazon EKS”. Para obter mais informações sobre como adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criar um complemento”. Se você não conseguir usar o complemento do Amazon EKS, recomendamos que você envie um problema sobre o motivo pelo qual não pode usar o repositório GitHub para roteiro de contêineres.

O complemento `kube-proxy` é implantado em cada nó do Amazon EC2 em seu cluster do Amazon EKS. Ele mantém as regras de rede em seus nós e permite a comunicação de rede com seus Pods. O complemento não é implantado em nós do Fargate em seu cluster. Para obter mais informações, consulte a `kube-proxy` documentação do Kubernetes.

A tabela a seguir lista a versão mais recente do tipo de complemento do Amazon EKS para cada versão do Kubernetes.

Versão do Kubernetes	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
	v1.30.0-eksbuild	v1.29.3-eksbuild	v1.28.8-eksbuild	v1.27.16-eksbuild.5	v1.26.15-eksbuild.5	v1.25.16-eksbuild.8	v1.24.17-eksbuild.8	v1.23.17-eksbuild.9

> **Important**
>
> Uma versão anterior da documentação estava incorreta. As versões `v1.28.5`, `v1.27.9` e `v1.26.12` do `kube-proxy` não estão disponíveis.
> Se você estiver gerenciando esse complemento automaticamente, as versões na tabela podem não ser as mesmas que as versões autogerenciadas disponíveis.

Há dois tipos de imagem de contêiner `kube-proxy` disponíveis para cada versão do cluster do Amazon EKS:

- Padrão: esse tipo de imagem é baseado em uma imagem do Docker baseado em Debian que é mantida pela comunidade do Kubernetes upstream.
- Mínima: esse tipo de imagem é baseado em uma imagem de base mínima mantida pelo Amazon EKS Distro, que contém pacotes mínimos e não tem shells. Para obter mais informações, consulte Amazon EKS Distro.

Versão mais recente da imagem do contêiner **`kube-proxy`** autogerenciado disponível para cada versão de cluster do Amazon EKS

Tipo de imagem	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
`kube-proxy` (tipo padrão)	Somente o tipo mínimo está	Somente o tipo mínimo está	Somente o tipo mínimo está	Somente o tipo mínimo está	Somente o tipo mínimo está	Somente o tipo mínimo está	v1.24.17-eksbuild.2	v1.23.16-eksbuild.2

Tipo de imagem	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
	disponíve l	disponíve l	disponíve l	disponíve l	disponíve l	disponíve l		
kube-proxy (tipo mínimo)	v1.30.0 m inimal- ek sbuild.	v1.29.3 m inimal- ek sbuild.	v1.28.8 m inimal- ek sbuild.	v1.27.1 minimal e ksbuild	v1.26.1 minimal e ksbuild	v1.25.1 minimal e ksbuild	v1.24.1 minimal e ksbuild	v1.23.17- minimal- e ksbuild.5

Important

- O tipo de imagem padrão não está disponível para a Kubernetes versão `1.25` e versões posteriores. Você deve usar o tipo mínimo de imagem.
- Ao atualizar um tipo de complemento do Amazon EKS, você especifica uma versão válida do complemento do Amazon EKS, que pode não ser uma versão listada nesta tabela. Isso ocorre porque as versões complementares do Amazon EKS nem sempre correspondem às versões de imagem de contêiner especificadas ao atualizar o tipo autogerenciado desse complemento. Ao atualizar o tipo autogerenciado desse complemento, você especifica uma versão de imagem de contêiner válida listada nesta tabela.

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS.

Considerações

- O `kube-proxy` em um cluster do Amazon EKS tem a mesma política de compatibilidade e distorção que o Kubernetes. Aprenda como Verificar a compatibilidade da versão do complemento do Amazon EKS com um cluster.

Para atualizar o complemento autogerenciado **kube-proxy**

1. Confirme que tem o tipo autogerenciado de complemento instalado em seu cluster. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
 addon.addonVersion --output text
```

   Se receber uma mensagem de erro, você tem o tipo autogerenciado do complemento instalado no cluster. As etapas restantes desse tópico servem para atualizar o tipo autogerenciado do complemento. Se receber um número de versão, você tem o tipo de complemento do Amazon EKS instalado no cluster. Para atualizá-lo, use o procedimento em Atualizar um complemento do Amazon EKS, em vez de usar o procedimento neste tópico. Se não estiver familiarizado com a diferença entre os tipos de complemento, consulte Complementos do Amazon EKS.

2. Veja qual versão da imagem do contêiner está atualmente instalada em seu cluster.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image
```

   Veja um exemplo de saída abaixo.

```
Image:    602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.29.1-
eksbuild.2
```

   No resultado de exemplo, *v1.29.1-eksbuild.2* é a versão instalada no cluster.

3. Atualize o complemento kube-proxy substituindo *602401143452* e *region-code* pelos valores dos resultados na etapa anterior. Substitua *v1.30.0-eksbuild.3* pela versão do kube-proxy listada na tabela Versão da imagem mais recente do contêiner kube-proxy disponível para cada versão de cluster do Amazon EKS. É possível especificar um número de versão para o tipo de imagem default (padrão) minimal (mínimo).

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-
proxy=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.30.0-
eksbuild.3
```

   Veja um exemplo de saída abaixo.

```
daemonset.apps/kube-proxy image updated
```

4. Confirme se agora a nova versão está instalada em seu cluster.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

Veja um exemplo de saída abaixo.

```
v1.30.0-eksbuild.3
```

5. Se você estiver usando os nós `x86` e `Arm` no mesmo cluster e o cluster tiver sido implantado antes de 17 de agosto de 2020. Em seguida, edite seu manifesto `kube-proxy` para incluir um seletor de nó para várias arquiteturas de hardware com o seguinte comando. É uma operação única. Depois que você adicionar o seletor ao manifesto, não precisará fazer isso toda vez que atualizar o complemento. Se o cluster foi implantado em ou após 17 de agosto de 2020, `kube-proxy` já é capaz de multiarquitetura.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Adicione o seletor de nós a seguir ao arquivo no editor e, em seguida, salve o arquivo. Para obter um exemplo de onde incluir esse texto no editor, consulte o arquivo CNI manifest (Manifesto de CNI) no GitHub. Isso permite que o Kubernetes extraia a imagem de hardware correta com base na arquitetura de hardware do nó.

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

6. Se o cluster tiver sido originalmente criado com o Kubernetes versão `1.14` ou posterior, você pode pular esta etapa, pois o `kube-proxy` já inclui essa `Affinity Rule`. Se você criou originalmente um cluster do Amazon EKS com o Kubernetes versão `1.13` ou anterior e pretender usar nós do Fargate no cluster, edite o manifesto do `kube-proxy` para incluir uma regra de `NodeAffinity` para evitar o agendamento de Pods do `kube-proxy` em nós do Fargate. Esta é uma edição ocasional. Depois que você adicionar o `Affinity Rule` ao manifesto, não precisará fazer isso toda vez que atualizar o complemento. Editar seu `kube-proxy DaemonSet`.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Adicione a seguinte `Affinity Rule` ao `DaemonSet` na seção `spec` do arquivo no editor e, em seguida, salve o arquivo. Para obter um exemplo de onde incluir esse texto no editor, consulte o arquivo CNI manifest (Manifesto de CNI) no GitHub.

```
- key: eks.amazonaws.com/compute-type
  operator: NotIn
  values:
  - fargate
```

Acesse o Amazon Elastic Kubernetes Service usando um endpoint de interface (AWS PrivateLink)

Você pode usar o AWS PrivateLink para criar uma conexão privada entre a sua VPC e o Amazon Elastic Kubernetes Service. Você pode acessar o Amazon EKS como se estivesse em sua VPC, sem usar um gateway da Internet, um dispositivo NAT, uma conexão VPN ou uma conexão AWS Direct Connect. As instâncias na VPC não precisam de endereços IP públicos para acessar o Amazon EKS.

Você estabelece essa conexão privada criando um endpoint de interface com a tecnologia AWS PrivateLink. Criaremos um endpoint de interface de rede em cada sub-rede que você habilitar para o endpoint de interface. Essas são interfaces de rede gerenciadas pelo solicitante que servem como ponto de entrada para o tráfego destinado ao Amazon EKS.

Para obter mais informações, consulte Acessar os Serviços da AWS pelo AWS PrivateLink no Guia do AWS PrivateLink.

Considerações sobre o Amazon EKS

- Antes de configurar um endpoint de interface para o Amazon EKS, analise as considerações no Guia do AWS PrivateLink.
- O Amazon EKS oferece suporte a chamadas para todas as ações de através do endpoint da interface, mas não para as APIs Kubernetes. O servidor de API Kubernetes já é compatível com um endpoint privado. O endpoint privado do servidor de API Kubernetes cria um endpoint privado para o servidor de API do Kubernetes usado para se comunicar com o cluster (usando as ferramentas de gerenciamento do Kubernetes, como o `kubectl`). Você pode habilitar o acesso privado ao servidor de API do Kubernetes de forma que todas as comunicações entre os nós e

o servidor de API permaneçam na VPC. O AWS PrivateLink para a API do Amazon EKS ajuda a chamar as APIs do Amazon EKS a partir da VPC sem expor o tráfego à Internet pública.

- Você não pode configurar o Amazon EKS para ser acessado somente por meio de um endpoint de interface.
- O preço padrão para o AWS PrivateLink aplica-se aos endpoints de interface para o Amazon EKS. Você é cobrado por cada hora que um endpoint de interface é provisionado em cada zona de disponibilidade e pelos dados processados por meio do endpoint de interface. Para obter mais informações, consulte Preços do AWS PrivateLink.
- As políticas de endpoint da VPC não são compatíveis com o Amazon EKS. Por padrão, acesso total ao Amazon EKS é permitido por meio do endpoint de interface. Ou então, você pode associar um grupo de segurança às interfaces de rede de endpoint para controlar o tráfego para o Amazon EKS por meio do endpoint de interface.
- Você pode usar os logs de fluxo da VPC para capturar informações sobre tráfego IP de entrada e de saída nas interfaces da rede, inclusive endpoints. Você pode publicar os dados do log de fluxo no Amazon CloudWatch Logs ou no Amazon S3. Para obter mais informações, consulte Como registrar tráfego IP em log com logs de fluxo da VPC no Manual do usuário da Amazon VPC.
- Você pode acessar as APIs do Amazon EKS de um data center on-premises conectando-o a uma VPC que tenha um endpoint de interface. Você pode usar o AWS Direct Connect ou AWS Site-to-Site VPN para conectar seus sites on-premises a uma VPC.
- Você pode conectar outras VPCs à VCP com um endpoint de interface usando um AWS Transit Gateway ou o emparelhamento da VPC. O emparelhamento de VPC é uma conexão de rede entre duas VPCs. Você pode estabelecer uma conexão de emparelhamento da VPC entre as suas VPCs ou com uma VPC de outra conta. As VPCs podem se encontrar em diferentes Regiões da AWS. O tráfego entre as VPCs emparelhadas permanece na rede da AWS. O tráfego não passa pela internet pública. Um gateway de trânsito é um hub de trânsito de rede que pode ser usado para interconectar as VPCs. O tráfego entre uma VPC e um gateway de trânsito permanece na rede privada global da AWS. O tráfego não é exposto à internet pública.
- Antes de agosto de 2024, os endpoints da interface da VPC para o Amazon EKS só eram acessíveis via `IPv4` com o uso de `eks.region.amazonaws.com`. Os novos endpoints de interface da VPC criados após agosto de 2024 usam pilha dupla de endereços IP `IPv4` e `IPv6` e ambos os nomes DNS: `eks.region.amazonaws.com` e `eks.region.api.aws`.

Criar um endpoint de interface de VPC para o Amazon EKS

Você pode criar um endpoint de interface para o Amazon EKS usando o console ou a AWS Command Line Interface (AWS CLI) da Amazon VPC. Para obter mais informações, consulte Create a VPC endpoint (Criar um endpoint da VPC) no Guia do AWS PrivateLink.

Crie um endpoint de interface para o Amazon EKS usando os seguintes nomes de serviço:

- `com.amazonaws.region-code.eks`
- `com.amazonaws.region-code.eks-auth`

O recurso de DNS privado é habilitado por padrão na criação de um endpoint de interface para o Amazon EKS e outros Serviços da AWS. Para usar o recurso de DNS privado, é necessário garantir que os seguintes atributos da VPC esteja definidos como `true`: `enableDnsHostnames` e `enableDnsSupport`. Para mais informações, consulte Visualizar e atualizar atributos DNS para sua VPC no Manual do usuário da Amazon VPC. Com o recurso de DNS privado ativado para o endpoint da interface:

- você pode fazer qualquer solicitação de API para o Amazon EKS usando seu nome DNS regional padrão. Depois de agosto de 2024, qualquer novo endpoint de interface VPC para a API do Amazon EKS terá dois nomes DNS regionais padrão e você poderá escolher `dualstack` para o tipo de endereço IP. O primeiro nome DNS é `eks.region.api.aws`, o qual é dual-stack. Ele resolve tanto endereços `IPv4` quanto endereços `IPv6`. Antes de agosto de 2024, o Amazon EKS usava apenas `eks.region.amazonaws.com` o que era resolvido somente em endereços `IPv4`. Se você quiser usar `IPv6` e empilhar dois endereços IP com um endpoint de interface da VPC existente, é possível atualizar o endpoint para usar o tipo de endereço IP `dualstack`, mas ele só terá o nome DNS `eks.region.amazonaws.com`. Nessa configuração, o endpoint existente é atualizado para apontar esse nome para ambos os endereços IP `IPv4` e `IPv6`. Para obter uma lista de APIs, consulte a Ações na Referência da API do Amazon EKS.
- Você não precisa fazer nenhuma alteração em suas aplicações que chamam as APIs do EKS.

  No entanto, para usar os endpoints de pilha dupla com a AWS CLI, consulte a configuração dos Endpoints de pilha dupla e FIPS no Guia de referência de SDKs e ferramentas da AWS.
- Qualquer chamada feita para o endpoint de serviço padrão do Amazon EKS é automaticamente roteada pelo endpoint de interface pela rede privada da AWS .

Saiba como implantar workloads e complementos no Amazon EKS

As workloads são implantadas em contêineres, que são implantados em Pods no Kubernetes. Um Pod inclui um ou mais contêineres. Normalmente, um ou mais Pods que fornecem o mesmo serviço são implantados em um serviço do Kubernetes. Depois de implantar vários Pods que fornecem o mesmo serviço, você poderá fazer as ações que se seguem.

- Visualizar informações sobre as workloads executadas em cada um dos clusters usando o AWS Management Console.
- Aumentar ou reduzir verticalmente a escala dos Pods com o Ajustar os recursos de pods com o Vertical Pod Autoscaler do Kubernetes.
- Aumentar ou reduza horizontalmente o número de Pods necessários para atender à demanda com o Escalar as implantações de pods com o Horizontal Pod Autoscaler do Kubernetes.
- Criar um balanceador de carga externo (para Pods acessíveis pela Internet) ou interno (para Pods privados) a fim de equilibrar o tráfego de rede entre Pods. O balanceador de carga direciona o tráfego na Camada 4 do modelo OSI.
- Criar um Aplicação de roteamento e tráfego HTTP com Application Load Balancers para balancear o tráfego de aplicações entre Pods. O balanceador de carga de aplicações direciona o tráfego na Camada 7 do modelo OSI.
- Se você não tiver experiência com o Kubernetes, este tópico ajudará você a Implantar uma aplicação de exemplo.
- Você pode restringir endereços IP que podem ser atribuídos a um serviço com `externalIPs`.

Implantar uma aplicação de exemplo

Neste tópico, você implanta uma aplicação de exemplo no cluster.

Pré-requisitos

- Um cluster Kubernetes existente com pelo menos um nó. Se você não tiver um cluster do Amazon EKS, implante um usando um dos guias Começar a usar o Amazon EKS. Se você estiver implantando uma aplicação do Windows, será necessário ter o suporte ao Windows habilitado para o cluster e pelo menos um nó do Windows do Amazon EC2.

- O `Kubectl` instalado em seu computador. Para ter mais informações, consulte Configurar o `kubectl` e o `eksctl`.
- O `Kubectl` configurado para se comunicar com o cluster. Para ter mais informações, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig.
- Se você planeja implantar sua workload de exemplo no Fargate, deverá ter um perfil do Fargate que inclua o mesmo namespace criado neste tutorial, que será `eks-sample-app`, a menos que você altere o nome. Se você usou um dos guias de conceitos básicos para criar o cluster, será necessário criar um novo perfil ou adicionar o namespace ao perfil existente, pois o perfil criado nos guias de introdução não especifica o namespace usado neste tutorial. A VPC também deve conter pelo menos uma sub-rede privada.

Para implantar uma aplicação de exemplo

Embora muitas variáveis possam ser alteradas nas etapas a seguir, recomendamos alterar apenas os valores das variáveis quando especificado. Depois que tiver entendido melhor os Pods, as implantações e os serviços do Kubernetes, você poderá experimentar alterar outros valores.

1. Crie um namespace . Um namespace permite agrupar recursos no Kubernetes. Para obter mais informações, consulte Namespaces na documentação do Kubernetes. Se você planeja implantar sua aplicação de exemplo para Simplificar o gerenciamento da computação com o AWS Fargate, verifique se o valor do `namespace` em Defina quais Pods usarão o AWS Fargate quando em execução é `eks-sample-app`.

   ```
   kubectl create namespace eks-sample-app
   ```

2. Crie um implantação do Kubernetes. Essa implantação de amostra extrai uma imagem de contêiner de um repositório público e implanta três réplicas (Pods individuais) dessa imagem no cluster. Para saber mais, consulte Deployments (Implantações) na documentação do Kubernetes. É possível implantar a aplicação em nós Linux ou Windows. Se estiver implantando no Fargate, só poderá implantar uma aplicação do Linux.

   a. Salve o conteúdo a seguir em um arquivo denominado `eks-sample-deployment.yaml`. Os contêineres da aplicação de exemplo não usam armazenamento em rede, mas pode haver aplicações que precisem. Para ter mais informações, consulte Armazenar dados de aplicações para seu cluster.

Linux

Os `values` de `amd64` ou `arm64` na chave `kubernetes.io/arch` significam que a aplicação pode ser implantada em qualquer arquitetura de hardware (se ambos estiverem no cluster). Isso é possível porque essa imagem é uma imagem multiarquitetura, mas nem todas são. É possível determinar a arquitetura de hardware em que há suporte para a imagem visualizando os detalhes da imagem no repositório do qual você está extraindo. Ao implantar imagens que não oferecem suporte a um tipo de arquitetura de hardware ou quando você não deseja que a imagem seja implantada, remova esse tipo do manifesto. Para obter mais informações, consulte Well-Known Labels, Annotations and Taints (Rótulos, anotações e taints conhecidos) na documentação do Kubernetes.

O `nodeSelector` do `kubernetes.io/os: linux` significa que se você tivesse (por exemplo) nós do Linux e do Windows no cluster, a imagem só seria implantada nos nós do Linux. Para obter mais informações, consulte Well-Known Labels, Annotations and Taints (Rótulos, anotações e taints conhecidos) na documentação do Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-linux-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-linux-app
  template:
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
```

```
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
      nodeSelector:
        kubernetes.io/os: linux
```

Windows

O `nodeSelector` do `kubernetes.io/os: windows` significa que se você tivesse (por exemplo) nós do Windows e do Linux no cluster, a imagem só seria implantada nos nós do Windows. Para obter mais informações, consulte Well-Known Labels, Annotations and Taints (Rótulos, anotações e taints conhecidos) na documentação do Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-windows-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-windows-app
  template:
    metadata:
      labels:
        app: eks-sample-windows-app
    spec:
      affinity:
        nodeAffinity:
```

```
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: beta.kubernetes.io/arch
                operator: In
                values:
                - amd64
      containers:
      - name: windows-server-iis
        image: mcr.microsoft.com/windows/servercore:ltsc2019
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
        command:
        - powershell.exe
        - -command
        - "Add-WindowsFeature Web-Server; Invoke-WebRequest -UseBasicParsing
 -Uri 'https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.6/
ServiceMonitor.exe' -OutFile 'C:\\ServiceMonitor.exe'; echo
 '<html><body><br/><br/><marquee><H1>Hello EKS!!!<H1><marquee></body><html>'
 > C:\\inetpub\\wwwroot\\default.html; C:\\ServiceMonitor.exe 'w3svc'; "
      nodeSelector:
        kubernetes.io/os: windows
```

b. Aplique o manifesto de implantação ao seu cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

3. Crie um serviço. Um serviço permite que você acesse todas as réplicas por meio de um único endereço IP ou nome. Para obter mais informações, consulte Service (Serviço) na documentação do Kubernetes. Embora não implementado na aplicação de amostra, se você tiver aplicações que precisem interagir com outros serviços da AWS, recomendamos criar contas de serviço do Kubernetes para os Pods e associá-las a contas do AWS IAM. Especificando contas de serviço, os Pods só têm as permissões mínimas que você especifica para interagir com outros serviços. Para ter mais informações, consulte Perfis do IAM para contas de serviço.

   a. Salve o conteúdo a seguir no arquivo denominado `eks-sample-service.yaml`. O Kubernetes atribui ao serviço o seu próprio endereço IP que só pode ser acessado de dentro do cluster. Para acessar o serviço de fora do seu cluster, implante o AWS Load

Balancer Controller para balancear a carga do tráfego de aplicações ou redes para o serviço.

Linux

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-linux-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  selector:
    app: eks-sample-linux-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Windows

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-windows-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  selector:
    app: eks-sample-windows-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

b. Aplique o manifesto de serviço ao seu cluster.

```
kubectl apply -f eks-sample-service.yaml
```

4. Visualize todos os recursos no namespace `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

Veja um exemplo de saída abaixo.

Se você implantou recursos do Windows, todas as instâncias do *linux* na saída a seguir serão windows. Os outros *exemplos de valores* podem ser diferentes da saída.

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/eks-sample-linux-deployment-65b7669776-m6qxz     1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-mmxvd     1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-qzn22     1/1     Running   0          27m

NAME                                  TYPE         CLUSTER-IP      EXTERNAL-IP
 PORT(S)   AGE
service/eks-sample-linux-service   ClusterIP    10.100.74.8     <none>        80/
TCP    32m

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-linux-deployment 3/3     3            3           27m

NAME                                                        DESIRED   CURRENT   READY
   AGE
replicaset.apps/eks-sample-linux-deployment-776d8f8fd8    3         3         3
   27m
```

Na saída, você verá o serviço e a implantação especificados nos manifestos de exemplo implantados nas etapas anteriores. Você também vê três Pods. Isso ocorre porque 3 `replicas` foram especificadas no manifesto de exemplo. Para obter mais informações sobre Pods, consulte Pod na documentação do Kubernetes. O Kubernetes cria automaticamente o arquivo do recurso `replicaset`, mesmo que não esteja especificado nos manifestos de exemplo. Para obter mais informações sobre `ReplicaSets`, consulte ReplicaSet na documentação do Kubernetes.

> **Note**
>
> O Kubernetes mantém o número de réplicas especificado no manifesto. Se essa implantação fosse uma implantação de produção e você quisesse que o Kubernetesescalasse horizontalmente o número de réplicas ou escalasse verticalmente os recursos de computação para os Pods, você usaria Escalar as implantações de pods

com o Horizontal Pod Autoscaler e Ajustar os recursos de pods com o Vertical Pod Autoscaler para fazer isso.

5. Veja os detalhes do serviço implantado. Se você implantou um serviço do Windows, substitua *linux* por **windows**.

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

Veja um exemplo de saída abaixo.

Se você implantou recursos do Windows, todas as instâncias do *linux* na saída a seguir serão windows. Os outros *exemplos de valores* podem ser diferentes da saída.

```
Name:              eks-sample-linux-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-linux-app
Annotations:       <none>
Selector:          app=eks-sample-linux-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

Na saída anterior, o valor para `IP:` é um endereço IP exclusivo que pode ser alcançado de qualquer nó ou Pod dentro do cluster, mas não pode ser alcançado de fora do cluster. Os valores para `Endpoints` são endereços IP atribuídos de dentro da VPC aos Pods que fazem parte do serviço.

6. Visualize os detalhes de um dos Pods listados na saída quando você visualizou o namespace em uma etapa anterior. Se você implantou uma aplicação do Windows, substitua *linux* por **windows** e substitua *776d8f8fd8-78w66* pelo valor retornado para um dos Pods.

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

Resultado abreviado

Se você implantou recursos do Windows, todas as instâncias do *linux* na saída a seguir serão windows. Os outros *example values* podem ser diferentes da saída.

```
Name:         eks-sample-linux-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-linux-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-linux-deployment-65b7669776-m6qxz
 to ip-192-168-45-132.us-west-2.compute.internal
[...]
```

Na saída anterior, o valor de `IP:` é um IP exclusivo atribuído ao Pod do bloco CIDR atribuído à sub-rede em que está o nó. Se preferir atribuir aos Pods endereços IP de blocos CIDR diferentes, você pode alterar o comportamento padrão. Para ter mais informações, consulte Rede personalizada para pods. Você também pode ver que o agendador do Kubernetes agendou o Pod no Node com o endereço IP *192.168.45.132*.

> **Tip**
>
> Em vez de usar a linha de comando, você pode visualizar muitos detalhes sobre Pods, serviços, implantações e outros recursos do Kubernetes no AWS Management Console. Para ter mais informações, consulte Visualizar os recursos do Kubernetes.

7. Execute um shell no Pod que você descreveu na etapa anterior, substituindo *65b7669776-m6qxz* pelo ID de um dos Pods.

   Linux

   ```
   kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app
    -- /bin/bash
   ```

   Windows

   ```
   kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-
   app -- powershell.exe
   ```

8. No shell do Pod, visualize a saída do servidor da Web que foi instalado com a implantação em uma etapa anterior. Basta especificar o nome do serviço. Ele é resolvido para o endereço IP do serviço pelo CoreDNS, que é implantado com um cluster do Amazon EKS, por padrão.

   Linux

   ```
   curl eks-sample-linux-service
   ```

   Veja um exemplo de saída abaixo.

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```

   Windows

   ```
   Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
   ```

Veja um exemplo de saída abaixo.

```
StatusCode        : 200
StatusDescription : OK
Content           : < h t m l > < b o d y > < b r / > < b r / > < m a r q u e e
 > < H 1 > H e l l o
                        E K S ! ! ! < H 1 > < m a r q u e e > < / b o d y > < h t
 m l >
```

9. No shell do Pod, visualize o servidor DNS para o Pod.

Linux

```
cat /etc/resolv.conf
```

Veja um exemplo de saída abaixo.

```
nameserver 10.100.0.10
search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local us-
west-2.compute.internal
options ndots:5
```

Na saída anterior, `10.100.0.10` é atribuído automaticamente como `nameserver` a todos os Pods implantados no cluster.

Windows

```
Get-NetIPConfiguration
```

Resultado abreviado

```
InterfaceAlias       : vEthernet
[...]
IPv4Address          : 192.168.63.14
[...]
DNSServer            : 10.100.0.10
```

Na saída anterior, `10.100.0.10` é atribuído automaticamente como o servidor de DNS a todos os Pods implantados no cluster.

10. Desconecte-se do Pod digitando `exit`.
11. Ao terminar de usar a aplicação de exemplo, você poderá remover o namespace, o serviço e a implantação de exemplo com o comando a seguir.

```
kubectl delete namespace eks-sample-app
```

Próximos Passos

Após implantar a aplicação de exemplo, talvez você queira tentar alguns dos seguintes exercícios:

- the section called “Balanceador de carga da aplicação”
- the section called “Balanceamento de carga da rede”

Ajustar os recursos de pods com o Vertical Pod Autoscaler

O Vertical Pod Autoscaler do Kubernetes ajusta automaticamente as reservas de CPU e de memória para os Pods, ajudando a "dimensionar corretamente" as aplicações. Esse ajuste pode melhorar a utilização de recursos do cluster e liberar CPU e memória para outros Pods. Este tópico ajuda você a implantar o Vertical Pod Autoscaler no cluster e verificar se ele está funcionando.

Pré-requisitos

- Você tem um cluster existente do Amazon EKS. Caso contrário, consulte Começar a usar o Amazon EKS.
- Você tem o servidor de métricas do Kubernetes instalado. Para ter mais informações, consulte Visualizar o uso de recursos com o KubernetesMetrics Server.
- Você estiver usando um cliente do `kubectl` que está configurado para se comunicar com o cluster do Amazon EKS.
- O OpenSSL `1.1.1` ou posterior instalado no seu dispositivo.

Implantar o Vertical Pod Autoscaler

Nesta seção, você implantará o Vertical Pod Autoscaler em seu cluster.

Como implantar o Vertical Pod Autoscaler

1. Abra uma janela do terminal e navegue até o diretório onde você deseja fazer download do código-fonte do Vertical Pod Autoscaler.
2. Clone o repositório kubernetes/autoscaler do GitHub.

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. Mude para o diretório `vertical-pod-autoscaler`.

```
cd autoscaler/vertical-pod-autoscaler/
```

4. (Opcional) Se você já tiver implantado outra versão do Vertical Pod Autoscaler, remova-a com o comando a seguir.

```
./hack/vpa-down.sh
```

5. Se os nós não tiverem acesso à Internet para o registro do contêiner `registry.k8s.io`, você precisará extrair as seguintes imagens e enviá-las para seu próprio repositório privado. Para obter mais informações sobre como extrair as imagens e enviá-las para seu próprio repositório privado, consulte Copiar uma imagem de contêiner de um repositório para outro.

```
registry.k8s.io/autoscaling/vpa-admission-controller:0.10.0
registry.k8s.io/autoscaling/vpa-recommender:0.10.0
registry.k8s.io/autoscaling/vpa-updater:0.10.0
```

   Se você estiver enviando as imagens para um repositório privado do Amazon ECR, substitua `registry.k8s.io` nos manifestos pelo seu registro. Substitua *`111122223333`* pelo ID da sua conta. Substitua *`region-code`* pela Região da AWS em que está o cluster. Para usar os comandos a seguir, você precisa ter nomeado seu repositório com o mesmo nome do repositório no manifesto. Se você nomeou o seu repositório como algo diferente, será necessário alterá-lo também.

```
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/recommender-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/updater-deployment.yaml
```

6. Implante o Vertical Pod Autoscaler no cluster com o comando a seguir.

```
./hack/vpa-up.sh
```

7. Verifique se os Pods do Vertical Pod Autoscaler foram criados com êxito.

```
kubectl get pods -n kube-system
```

Veja um exemplo de saída abaixo.

```
NAME                                        READY   STATUS    RESTARTS   AGE
[...]
metrics-server-8459fc497-kfj8w              1/1     Running   0          83m
vpa-admission-controller-68c748777d-ppspd   1/1     Running   0          7s
vpa-recommender-6fc8c67d85-gljpl            1/1     Running   0          8s
vpa-updater-786b96955c-bgp9d                1/1     Running   0          8s
```

Testar a instalação do Vertical Pod Autoscaler

Nesta seção, você implantará uma aplicação de exemplo para verificar se o Vertical Pod Autoscaler está funcionando.

Como testar a instalação do Vertical Pod Autoscaler

1. Implante o exemplo `hamster.yaml` do Vertical Pod Autoscaler com o comando a seguir.

```
kubectl apply -f examples/hamster.yaml
```

2. Obtenha os Pods da aplicação de exemplo do `hamster`.

```
kubectl get pods -l app=hamster
```

Veja um exemplo de saída abaixo.

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. Descreva um dos Pods para visualizar a reserva de `cpu` e `memory`. Substitua *`c7d89d6db-rglf5`* por um dos IDs retornados na saída da etapa anterior.

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

Veja um exemplo de saída abaixo.

```
[...]
Containers:
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:35:16 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     50Mi
[...]
```

Você pode ver que o Pod original reserva 100 milicpu de CPU e 50 mebibytes de memória. Para esta aplicação de exemplo, 100 milicpu é menos do que o Pod precisa para ser executado, portanto, ele tem restrição de CPU. Ele também reserva muito menos memória do que precisa. A implantação `vpa-recommender` do Vertical Pod Autoscaler analisa os Pods `hamster` para ver se os requisitos de CPU e memória são apropriados. Se ajustes forem necessários, o `vpa-updater` reiniciará os Pods com valores atualizados.

4. Aguarde até que o `vpa-updater` inicie e um novo Pod `hamster`. Isso deve levar um ou dois minutos. É possível monitorar os Pods com o comando a seguir.

> **Note**
>
> Se você não tiver certeza de que um novo Pod foi iniciado, compare os nomes de Pod com a lista anterior. Quando o novo Pod for iniciado, você verá um novo nome de Pod.

```
kubectl get --watch Pods -l app=hamster
```

5. Quando um novo Pod `hamster` for iniciado, descreva-o e visualize as reservas atualizadas de CPU e memória.

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

Veja um exemplo de saída abaixo.

```
[...]
Containers:
  hamster:
    Container ID:
 docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:37:08 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        587m
      memory:     262144k
[...]
```

Na saída anterior, você pode ver que a reserva de `cpu` e aumentou para 587 milicpu, que é mais de cinco vezes o valor original. A `memory` aumentou para 262.144 Kilobytes, que é cerca de 250 mebibytes, ou cinco vezes o valor original. Esse Pod estava com poucos recursos, e o Vertical Pod Autoscaler corrigiu a estimativa com um valor muito mais apropriado.

6. Descreva o recurso `hamster-vpa` para visualizar a nova recomendação.

```
kubectl describe vpa/hamster-vpa
```

Veja um exemplo de saída abaixo.

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/
v1beta2","kind":"VerticalPodAutoscaler","metadata":{"annotations":
{},"name":"hamster-vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:          23
  Resource Version:    14411
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/
verticalpodautoscalers/hamster-vpa
  UID:                 d0d85fb9-e153-11e9-ae53-0205785d75b0
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
  Conditions:
    Last Transition Time:  2019-09-27T18:23:28Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
        Cpu:     550m
```

```
        Memory:  262144k
      Target:
        Cpu:     587m
        Memory:  262144k
      Uncapped Target:
        Cpu:     587m
        Memory:  262144k
      Upper Bound:
        Cpu:     21147m
        Memory:  387863636
Events:          <none>
```

7. Depois de experimentar com a aplicação de exemplo, exclua-a com os comandos a seguir.

```
kubectl delete -f examples/hamster.yaml
```

Escalar as implantações de pods com o Horizontal Pod Autoscaler

O Horizontal Pod Autoscaler do Kubernetes dimensiona automaticamente o número de Pods em uma implantação, o controlador de replicação ou o conjunto de réplicas com base na utilização da CPU desse recurso. Isso pode ajudar a expandir as aplicações para atender ao aumento da demanda ou a reduzi-las quando os recursos não forem necessários, liberando os nós para outras aplicações. Quando você define uma porcentagem de utilização de CPU, o Horizontal Pod Autoscaler reduz ou expande a aplicação para tentar atingir essa meta.

O Horizontal Pod Autoscaler é um recurso de API padrão no Kubernetes que simplesmente requer que uma fonte de métricas (como o servidor de métricas do Kubernetes) esteja instalada no cluster do Amazon EKS para funcionar. Você não precisa implantar ou instalar o Horizontal Pod Autoscaler no cluster para começar a escalar as aplicações. Para obter mais informações, consulte a Horizontal Pod Autoscaler documentação do Kubernetes.

Use este tópico para preparar o Horizontal Pod Autoscaler para seu cluster do Amazon EKS e verificar se ele está funcionando com uma aplicação de exemplo.

> **Note**
>
> Este tópico se baseia na Demonstração do Horizontal Pod autoscaler, na documentação do Kubernetes.

Pré-requisitos

- Você tem um cluster existente do Amazon EKS. Caso contrário, consulte Começar a usar o Amazon EKS.
- Você tem o servidor de métricas do Kubernetes instalado. Para ter mais informações, consulte Visualizar o uso de recursos com o KubernetesMetrics Server.
- Você estiver usando um cliente do `kubectl` que está configurado para se comunicar com o cluster do Amazon EKS.

Execute uma aplicação de teste do Horizontal Pod Autoscaler

Nesta seção, você implanta uma aplicação de exemplo para verificar se o Horizontal Pod Autoscaler está funcionando.

> **Note**
>
> Este exemplo é baseado na Demonstração do Horizontal Pod Autoscaler, na documentação do Kubernetes.

Para testar a instalação do Horizontal Pod Autoscaler

1. Implante uma aplicação simples de servidor Web do Apache, com o comando a seguir.

   ```
   kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
   ```

   É atribuído um limite de CPU de 500 milicpu a esse Pod de servidor Web do Apache e ele está atendendo na porta 80.
2. Crie um recurso Horizontal Pod Autoscaler para a implantação do `php-apache`.

   ```
   kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
   ```

   Esse comando cria um autoscaler que tem como meta 50% de utilização de CPU para a implantação, com um mínimo de um Pod e um máximo de dez Pods. Quando a carga média da CPU for inferior a 50%, o autoscaler tentará reduzir o número de Pods na implantação para um mínimo de um. Quando a carga for maior que 50%, o autoscaler tentará aumentar o número de Pods na implantação, até um máximo de dez. Para obter mais informações, consulte How

does the Horizontal Pod Autoscaler work? (Funcionamento do Horizontal Pod Autoscaler) na documentação do Kubernetes.

3. Descreva o autoscaler com o seguinte comando para visualizar seus detalhes.

```
kubectl get hpa
```

Veja um exemplo de saída abaixo.

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   0%/50%    1         10        1          51s
```

Como você pode ver, a carga atual da CPU é `0%`, pois ainda não há nenhuma carga no servidor. O número de Pod já está no limite mais baixo (um), portanto, não pode ser reduzido.

4. Crie uma carga para o servidor web executando um contêiner.

```
kubectl run -i \
    --tty load-generator \
    --rm --image=busybox \
    --restart=Never \
    -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
```

5. Para observar a expansão da implantação, execute periodicamente o seguinte comando em um terminal separado do terminal em que você executou a etapa anterior.

```
kubectl get hpa php-apache
```

Veja um exemplo de saída abaixo.

```
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   250%/50%   1         10        5
 4m44s
```

A contagem de réplicas pode levar mais de um minuto para aumentar. Enquanto a porcentagem real da CPU for maior do que a porcentagem de meta, a contagem de réplicas aumentará, até 10. Neste caso, é `250%`, então o número de `REPLICAS` continua a aumentar.

> **Note**
>
> Poderá demorar alguns minutos para você ver a contagem de réplicas atingir seu máximo. Se apenas 6 réplicas, por exemplo, forem necessárias para que a carga da CPU permaneça em 50% ou menos, a carga não escalará além de 6 réplicas.

6. Pare a carga. Na janela do terminal em que você está gerando a carga, interrompa a carga mantendo pressionadas as teclas `Ctrl+C`. Você pode observar a escala de réplicas voltar para 1 executando o comando a seguir novamente no terminal em que está observando a redução da escala na horizontal.

```
kubectl get hpa
```

Veja um exemplo de saída abaixo.

```
NAME         REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache      0%/50%    1         10        1          25m
```

> **Note**
>
> O período de tempo padrão para reduzir a escala na vertical é de cinco minutos, portanto, levará algum tempo para você ver a contagem de réplicas chegar a 1 novamente, até mesmo quando a percentagem de CPU atual for 0%. O período de tempo é modificável. Para obter mais informações, consulte Horizontal Pod Autoscaler na documentação do Kubernetes.

7. Ao concluir os testes da aplicação de exemplo, exclua os recursos do `php-apache`.

```
kubectl delete deployment.apps/php-apache service/php-apache
 horizontalpodautoscaler.autoscaling/php-apache
```

Roteamento TCP e tráfego UDP com Network Load Balancers

O tráfego da rede tem a carga balanceada no L4 do modelo OSI. Para balancear a carga do tráfego da aplicação no L7, você implanta um Kubernetes `ingress`, que provisiona um AWS Application Load Balancer. Para ter mais informações, consulte Aplicação de roteamento e tráfego

HTTP com Application Load Balancers. Para saber mais sobre as diferenças entre os dois tipos de balanceamento de carga, consulte Elastic Load Balancing features (Recursos do Elastic Load Balancing) no site da AWS.

Ao criar um `Service` do Kubernetes do tipo `LoadBalancer`, o controlador do balanceador de carga de provedor da nuvem da AWS cria AWS Classic Load Balancers por padrão, mas também pode criar AWS Network Load Balancers. Este controlador só receberá correções de erros críticos no futuro. Para obter mais informações sobre como usar o balanceador de carga de provedor de nuvem da AWS, consulte Controlador do balanceador de carga de provedor de nuvem da AWS na documentação do Kubernetes. Seu uso não é abordado neste tópico.

Recomendamos que você use a versão `2.7.2` ou posterior do AWS Load Balancer Controller em vez do controlador do balanceador de carga do provedor de nuvem da AWS. O AWS Load Balancer Controller cria AWS Network Load Balancers, mas não cria AWS Classic Load Balancers. O restante deste tópico refere-se ao uso do AWS Load Balancer Controller.

Um AWS Network Load Balancer pode balancear a carga do tráfego de rede para Pods implantados em destinos de IP e de instâncias do Amazon EC2 ou para destinos IP do AWS Fargate. Para obter mais informações, consulte GitHub Load Balancer Controller (Controlador do AWS Load Balancer) no .

Pré-requisitos

Antes de balancear a carga de tráfego da rede do balanceador de carga usando o AWS Load Balancer Controller, você precisa cumprir os requisitos a seguir.

- Tem um cluster já existente. Se você não tem um cluster, consulte Começar a usar o Amazon EKS. Se precisar atualizar a versão de um cluster existente, consulte Atualizar um cluster existente para a nova versão do Kubernetes.
- Ter o AWS Load Balancer Controller implantado no seu cluster. Para ter mais informações, consulte O que é o AWS Load Balancer Controller?. Recomendamos a versão `2.7.2` ou posterior.
- Pelo menos uma sub-rede. Se várias sub-redes marcadas são encontradas em uma zona de disponibilidade, o controlador escolhe a primeira sub-rede por ordem lexicográfica de IDs. A sub-rede deve ter pelo menos oito endereços IP disponíveis.
- Se você estiver usando o AWS Load Balancer Controller versão `2.1.1` ou anterior, as sub-redes deverão ser marcadas do modo a seguir. Se estiver usando a versão `2.1.2` ou superior, essa etiqueta será opcional. Talvez seja bom marcar uma sub-rede se você tiver vários clusters em execução na mesma VPC ou várias sub-redes de compartilhamento de serviços da AWS em uma

VPC e quiser ter mais controle sobre onde os balanceadores de carga são provisionados por cluster. Se você especificar explicitamente os IDs de sub-redes como uma anotação em um objeto de serviço, o Kubernetes e o AWS Load Balancer Controller usarão essas sub-redes diretamente para criar o balanceador de carga. A marcação da sub-rede não é necessária se você optar por usar esse método para provisionar balanceadores de carga, e você pode ignorar os requisitos de marcação de sub-rede privada e pública a seguir. Substitua *my-cluster* pelo nome do cluster.

- Chave – `kubernetes.io/cluster/`*my-cluster*
- Valor: `shared` ou `owned`

- As suas sub-redes públicas e privadas devem atender aos requisitos a seguir, a menos que você especifique explicitamente os IDs de sub-rede como uma anotação em um objeto de serviço ou de entrada. Se você provisionar balanceadores de carga especificando explicitamente os IDs de sub-redes como uma anotação em um objeto de serviço ou ingresso, o Kubernetes e o AWS Load Balancer Controller usam essas sub-redes diretamente para criar o balanceador de carga, e as tags a seguir não serão necessárias.
  - Sub-redes privadas – Devem ser marcadas no seguinte formato. Isso permite que o Kubernetes e o Controlador do AWS Load Balancer saibam que as sub-redes podem ser usadas para balanceadores de carga internos. Se você usar o `eksctl` ou um modelo do AWS AWS CloudFormation no Amazon EKS para criar a VPC após 26 de março de 2020, as sub-redes serão marcadas adequadamente quando forem criadas. Para obter mais informações sobre os modelos de VPC do AWS AWS CloudFormation no Amazon EKS, consulte Criar uma VPC para o cluster do Amazon EKS.
    - Chave – `kubernetes.io/role/internal-elb`
    - Value (valor): 1
  - Sub-redes públicas – Devem ser marcadas no seguinte formato. Isso permite o Kubernetes saiba que só deve usar essas sub-redes para balanceadores de carga externos, em vez de escolher uma sub-rede pública em cada zona de disponibilidade (em ordem lexicográfica por ID de sub-rede). Se você usar o `eksctl` ou um modelo do AWS CloudFormation no Amazon EKS para criar a VPC após 26 de março de 2020, as sub-redes serão marcadas adequadamente quando forem criadas. Para obter mais informações sobre os modelos de VPC do AWS CloudFormation no Amazon EKS, consulte Criar uma VPC para o cluster do Amazon EKS.
    - Chave – `kubernetes.io/role/elb`
    - Value (valor): 1

Se tags de função de sub-rede não forem explicitamente adicionadas, o controlador de serviço do Kubernetes analisará a tabela de rotas das sub-redes da VPC do cluster para determinar se a sub-rede é privada ou pública. Recomendamos que você não confie nesse comportamento e adicione explicitamente as etiquetas de função privada ou pública. O AWS Load Balancer Controller não examina as tabelas de rotas e exige que as etiquetas privadas e públicas estejam presentes para a detecção automática ser bem-sucedida.

Considerações

- A configuração do balanceador de carga é controlada por anotações que são adicionadas ao manifesto para o serviço. As anotações de serviço são diferentes ao usar o AWS Load Balancer Controller e o controlador do balanceador de carga do provedor da nuvem AWS. Certifique-se de revisar as anotações para o AWS Load Balancer Controller antes de implantar os serviços.
- Ao usar o Amazon VPC CNI plugin for Kubernetes, o AWS Load Balancer Controller pode balancear a carga para destinos de IP ou de instância do Amazon EC2 e destinos de IP do Fargate. Ao usar Alternate compatible CNI plugins (Alternar plugins da CNI compatíveis), o controlador só poderá balancear a carga para destinos de instância. Para obter mais informações sobre os tipos de destino do Network Load Balancer, consulte Target type (Tipos de destino) no Guia do usuário de Network Load Balancers.
- Se quiser adicionar etiquetas ao balanceador de carga quando ou depois que ele for criado, adicione a anotação a seguir na especificação do seu serviço. Para obter mais informações, consulte Etiquetas de recurso da AWS na documentação do AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
```

- Você pode atribuir endereços IP elásticos ao Network Load Balancer adicionando a anotação a seguir. Substitua *example values* pelos `Allocation IDs` dos endereços de IP elásticos. O número de `Allocation IDs` deve corresponder ao número de sub-redes usadas para o balanceador de carga. Para obter mais informações, consulte a documentação do AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations:
 eipalloc-xxxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyyy
```

- O Amazon EKS adiciona uma regra de entrada ao grupo de segurança do nó para o tráfego de clientes e uma regra para cada sub-rede do balanceador de carga na VPC para as verificações de integridade para cada Network Load Balancer que você criar. A implantação de um serviço do tipo

`LoadBalancer` poderá falhar se o Amazon EKS tentar criar regras que excedam a cota para o número máximo de regras permitidas para um grupo de segurança. Para obter mais informações, consulte Security groups (Grupos de segurança) em Amazon VPC quotas (Cotas da Amazon VPC) no Manual do usuário da Amazon VPC. Considere as opções a seguir para minimizar as chances de exceder o número máximo de regras para um grupo de segurança:

- Solicite um aumento em suas regras por cota de grupo de segurança. Para obter mais informações, consulte Requesting a quota increase (Solicitar um aumento da cota) no Manual do usuário do Service Quotas.
- Use destinos IP, em vez de destinos instância. Com destinos IP, você pode compartilhar as regras para as mesmas portas de destino. As sub-redes do balanceador de carga podem ser especificadas manualmente com uma anotação. Para obter mais informações, consulte Annotations (Anotações) no GitHub.
- Use uma entrada em vez de um serviço do tipo `LoadBalancer` para enviar tráfego ao seu serviço. O AWS Application Load Balancer exige menos regras do que os Network Load Balancers. Você pode compartilhar um ALB em várias entradas. Para ter mais informações, consulte Aplicação de roteamento e tráfego HTTP com Application Load Balancers. Você não pode compartilhar um Network Load Balancer em vários serviços.
- Implante seus clusters em várias contas.

- Se os Pods forem executados no Windows em um cluster do Amazon EKS, um único serviço com um balanceador de carga poderá ser compatível com até 1024 Pods de backend. Cada Pod tem seu próprio endereço IP exclusivo.
- Recomendamos que você só crie novos Network Load Balancers com o AWS Load Balancer Controller. A tentativa de substituir Network Load Balancers existentes criados pelo controlador do balanceador de carga do provedor da nuvem AWS pode resultar em vários Network Load Balancers, que poderão gerar tempo de inatividade na aplicação.

Criar um balanceador de carga da rede

Você pode criar um balanceador de carga da rede com destinos de IP ou de instância.

IP targets

Use destinos IP com Pods implantados em nós do Amazon EC2 ou Fargate. O serviço do Kubernetes deve ser criado como tipo `LoadBalancer`. Para obter mais informações, consulte Type LoadBalancer (Tipo LoadBalancer) na documentação do Kubernetes.

Para criar um balanceador de carga que use destinos IP, adicione a anotação a seguir a um manifesto de serviço e implante o seu serviço. O valor `external` para `aws-load-balancer-type` faz com que o AWS Load Balancer Controller, em vez do controlador do balanceador de carga do provedor da nuvem AWS, crie o Network Load Balancer. Você pode visualizar um manifesto de serviço de exemplo com as anotações.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

> **Note**
>
> Se você estiver balanceando carga para Pods do `IPv6`, adicione a anotação a seguir. Você só pode balancear a carga em `IPv6` para destinos IP, e não para destinos de instâncias. Sem essa anotação, o balanceamento de carga ocorrerá em `IPv4`.
>
> ```
> service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
> ```

Por padrão, os Network Load Balancers são criados com o `aws-load-balancer-scheme` `internal`. Você pode iniciar Network Load Balancers em qualquer sub-rede na VPC do cluster, incluindo sub-redes que não foram especificadas quando o cluster foi criado.

O Kubernetes examina a tabela de rotas para as sub-redes a fim de identificar se elas são públicas ou privadas. As sub-redes públicas têm uma rota direta para a Internet usando um gateway da Internet, mas as sub-redes privadas não têm.

Se você quiser criar um Network Load Balancer em uma sub-rede pública para balancear a carga para nós do Amazon EC2 (o Fargate só pode ser em privadas), especifique `internet-facing` com a seguinte anotação:

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Note**
>
> Para fins de compatibilidade com versões anteriores, o suporte a anotações `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` ainda é oferecido. Porém, recomendamos usar as anotações anteriores para novos

balanceadores de carga em vez do `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"`.

> **Important**
>
> Não edite as anotações depois de criar o serviço. Se precisar modificá-lo, exclua o objeto de serviço e crie-o novamente com o valor desejado para essa anotação.

Instance targets

O controlador do balanceador de carga do provedor da nuvem AWS cria Network Load Balancers com apenas destinos de instâncias. A versão `2.2.0` e versões superiores do AWS Load Balancer Controller também criam Network Load Balancers com destinos de instâncias. Recomendamos usá-lo para criar novos Network Load Balancers, em vez do controlador do balanceador de carga do provedor da nuvem AWS. Você pode usar os destinos de instância do Network Load Balancer com Pods implantados em nós do Amazon EC2, mas não do Fargate. Para balancear a carga do tráfego de rede em Pods implantados no Fargate, você deve usar destinos IP.

Para implantar um Network Load Balancer em uma sub-rede privada, a sua especificação de serviço deve ter as anotações a seguir. Você pode visualizar um manifesto de serviço de exemplo com as anotações. O valor `external` para `aws-load-balancer-type` faz com que o AWS Load Balancer Controller, em vez do controlador do balanceador de carga do provedor da nuvem AWS, crie o Network Load Balancer.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
```

Por padrão, os Network Load Balancers são criados com o `aws-load-balancer-scheme` `internal`. Para Network Load Balancers internos, o cluster do Amazon EKS deve ser configurado para usar pelo menos uma sub-rede privada na VPC. O Kubernetes examina a tabela de rotas das sub-redes para identificar se elas são públicas ou privadas. As sub-redes públicas têm uma rota direta para a Internet usando um gateway da Internet, mas as sub-redes privadas não têm.

Se você quiser criar um Network Load Balancer em uma sub-rede pública para balancear a carga para nós do Amazon EC2, especifique `internet-facing` com a seguinte anotação:

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Important**
>
> Não edite as anotações depois de criar o serviço. Se precisar modificá-lo, exclua o objeto de serviço e crie-o novamente com o valor desejado para essa anotação.

(Opcional) Implantar uma aplicação de exemplo

Pré-requisitos

- Pelo menos uma sub-rede pública ou privada na VPC do cluster.
- Ter o AWS Load Balancer Controller implantado no seu cluster. Para ter mais informações, consulte O que é o AWS Load Balancer Controller?. Recomendamos a versão `2.7.2` ou posterior.

Para implantar uma aplicação de exemplo

1. Se estiver implantando no Fargate, certifique-se de ter uma sub-rede privada disponível em sua VPC e crie um perfil do Fargate. Se não estiver implantando no Fargate, ignore esta etapa. Você pode criar o perfil executando o comando a seguir ou no AWS Management Console usando os mesmos valores para `name` e `namespace` que estão no comando. Substitua *`example values`* pelos seus próprios valores.

   ```
   eksctl create fargateprofile \
       --cluster my-cluster \
       --region region-code \
       --name nlb-sample-app \
       --namespace nlb-sample-app
   ```

2. Implante uma aplicação de exemplo.

   a. Crie um namespace para a aplicação.

      ```
      kubectl create namespace nlb-sample-app
      ```

b. Salve o conteúdo a seguir em seu computador, em um arquivo chamado *sample-deployment*.yaml.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nlb-sample-app
  namespace: nlb-sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

c. Aplique o manifesto ao cluster.

```
kubectl apply -f sample-deployment.yaml
```

3. Crie um serviço com um Network Load Balancer voltado para a Internet que faça o balanceamento de carga para destinos IP.

a. Salve o conteúdo a seguir em seu computador, em um arquivo chamado *sample-service*.yaml. Se estiver implantando nos nós do Fargate, remova a linha `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing`.

```
apiVersion: v1
kind: Service
metadata:
  name: nlb-sample-service
  namespace: nlb-sample-app
```

```
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

b. Aplique o manifesto ao cluster.

```
kubectl apply -f sample-service.yaml
```

4. Verifique se o serviço foi implantado.

```
kubectl get svc nlb-sample-service -n nlb-sample-app
```

Veja um exemplo de saída abaixo.

```
NAME             TYPE           CLUSTER-IP         EXTERNAL-IP
                                                 PORT(S)        AGE
sample-service   LoadBalancer   10.100.240.137
 k8s-nlbsampl-nlbsampl-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
  80:32400/TCP   16h
```

> **Note**
>
> Os valores para *10.100.240.137* e *xxxxxxxxxx-xxxxxxxxxxxxxxxx* serão diferentes da saída do exemplo (eles serão exclusivos do seu balanceador de carga) e *us-west-2* podem ser diferentes para você, dependendo de qual Região da AWS for o seu cluster.

5. Abra o AWS Management Console do Amazon EC2. Selecione Target Groups (Grupos de destino) em Load Balancing (Balanceamento de carga) no painel de navegação à esquerda. Em Name (Nome), selecione o nome do grupo de destino no qual o valor na coluna Load balancer (Balanceador de carga) corresponda a uma parte do nome na coluna `EXTERNAL-IP`

do resultado na etapa anterior. Por exemplo, você selecionaria o grupo de destino chamado `k8s-default-samplese-xxxxxxxxxx` se o resultado fosse igual ao da saída acima. O tipo de destino é `IP`, conforme especificado no manifesto de serviço de exemplo.

6. Selecione o Target group (Grupo de destino) e escolha a guia Targets (Destinos). Em Registered targets (Destinos registrados), você verá três endereços IP das três réplicas implantadas em uma etapa anterior. Aguarde até que o status de todos os destinos seja healthy (íntegro) antes de continuar. Pode levar alguns minutos até que todos os destinos fiquem `healthy`. Os destinos podem estar em um estado `unhealthy` antes de serem alterados para um estado `healthy`.
7. Envie o tráfego para o serviço, substituindo *xxxxxxxxxx-xxxxxxxxxxxxxxxx* e *us-west-2* pelo valor retornado na saída de uma etapa anterior por `EXTERNAL-IP`. Se você implantou em uma sub-rede privada, precisará visualizar a página em um dispositivo dentro da VPC, como um bastion host. Para obter mais informações, consulte Bastion hosts Linux na AWS.

   ```
   curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
   ```

   Veja um exemplo de saída abaixo.

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```

8. Ao terminar de usar a implantação, o serviço e o namespace de exemplo, remova-os.

   ```
   kubectl delete namespace nlb-sample-app
   ```

Aplicação de roteamento e tráfego HTTP com Application Load Balancers

Quando você cria um `ingress` do Kubernetes, um AWS Application Load Balancer (ALB) é provisionado e equilibra a carga de tráfego da aplicação. Para saber mais, consulte What is an Application Load Balancer? (O que é um balanceador de carga de aplicação?) no Application Load Balancers (Guia do usuário de Application Load Balancers) e Ingress (Ingresso) na documentação do

Kubernetes. Os ALBs podem ser usados com Pods implantados em nós ou no AWS Fargate. Você pode implantar um ALB em sub-redes públicas ou privadas.

O tráfego de aplicações é equilibrado no L7 do modelo OSI. Para balancear a carga de tráfego de rede em L4, você implanta um `service` do Kubernetes do tipo `LoadBalancer`. Este tipo provisiona um tipo provisiona um AWS Network Load Balancer. Para ter mais informações, consulte Roteamento TCP e tráfego UDP com Network Load Balancers. Para saber mais sobre as diferenças entre os dois tipos de balanceamento de carga, consulte Elastic Load Balancing features (Recursos do Elastic Load Balancing) no site da AWS.

Pré-requisitos

Antes de balancear a carga de tráfego de uma aplicação específica, você precisa cumprir os requisitos a seguir.

- Tem um cluster já existente. Se você não tem um cluster, consulte Começar a usar o Amazon EKS. Se precisar atualizar a versão de um cluster existente, consulte Atualizar um cluster existente para a nova versão do Kubernetes.
- Ter o AWS Load Balancer Controller implantado no seu cluster. Para ter mais informações, consulte O que é o AWS Load Balancer Controller?. Recomendamos a versão `2.7.2` ou posterior.
- As duas sub-redes devem estar em diferentes zonas de disponibilidade. O AWS Load Balancer Controller escolhe uma sub-rede de cada zona de disponibilidade. Quando várias sub-redes marcadas são encontradas em uma zona de disponibilidade, o controlador escolhe a primeira sub-rede, por ordem lexicográfica de IDs. Cada sub-rede deve ter pelo menos oito endereços IP disponíveis.

  Se você estiver usando vários grupos de segurança anexados ao nó de processamento, exatamente um grupo de segurança deve ser marcado como se segue. Substitua *`my-cluster`* pelo nome do cluster.

  - Chave – `kubernetes.io/cluster/`*`my-cluster`*
  - Valor: `shared` ou `owned`
- Se você estiver usando o AWS Load Balancer Controller versão `2.1.1` ou anterior, as sub-redes deverão ser marcadas como se segue. Se estiver usando a versão `2.1.2` ou superior, essa marcação é opcional. Porém, recomendamos que você marque uma sub-rede em qualquer um dos seguintes casos. Você tiver vários clusters que estão sendo executados na mesma VPC ou tiver vários serviços da AWS que compartilham sub-redes em uma VPC. Ou, você quiser ter mais

controle sobre onde os balanceadores de carga são provisionados para cada cluster. Substitua *my-cluster* pelo nome do cluster.

- Chave – `kubernetes.io/cluster/`*my-cluster*
- Valor: `shared` ou `owned`

- As sub-redes públicas e privadas devem atender aos seguintes requisitos, Isso só não ocorre caso você especifique explicitamente IDs de sub-redes como uma anotação em um objeto de serviço ou entrada. Pressuponha que você provisione balanceadores de carga especificando explicitamente os IDs de sub-redes como uma anotação em um objeto de serviço ou entrada. Nessa situação, o Kubernetes e o AWS Load Balancer Controller usam as sub-redes diretamente para criar o balanceador de carga e as tags a seguir não são necessárias.
  - Sub-redes privadas – Devem ser marcadas no seguinte formato. Isso ocorre para que o Kubernetes e o AWS Load Balancer Controller saibam que as sub-redes podem ser usadas para balanceadores de carga internos. Se você usar o `eksctl` ou um modelo de AWS CloudFormation do Amazon EKS para criar a VPC após 26 de março de 2020, as sub-redes serão marcadas adequadamente quando forem criadas. Para obter mais informações sobre os modelos de VPC do AWS CloudFormation no Amazon EKS, consulte Criar uma VPC para o cluster do Amazon EKS.
    - Chave – `kubernetes.io/role/internal-elb`
    - Value (valor): 1
  - Sub-redes públicas – Devem ser marcadas no seguinte formato. Isso ocorre para que o Kubernetes saiba que só deve usar as sub-redes especificadas para balanceadores de carga externos. Dessa forma, o Kubernetes não escolhe uma sub-rede pública em cada zona de disponibilidade (por ordem lexicográfica dos IDs de sub-rede). Se você usar o `eksctl` ou um modelo de AWS CloudFormation do Amazon EKS para criar a VPC após 26 de março de 2020, as sub-redes serão marcadas adequadamente quando forem criadas. Para obter mais informações sobre os modelos de VPC do AWS CloudFormation no Amazon EKS, consulte Criar uma VPC para o cluster do Amazon EKS.
    - Chave – `kubernetes.io/role/elb`
    - Value (valor): 1

  Se as tags de função de sub-rede não forem explicitamente adicionadas, o controlador de serviço do Kubernetes analisará a tabela de rotas das sub-redes da VPC do cluster. Isso ocorre para determinar se a sub-rede é privada ou pública. Recomendamos que você não confie nesse comportamento. Em vez disso, adicione explicitamente as etiquetas de função privada ou pública.

O AWS Load Balancer Controller não examina tabelas de rotas. Também requer que as etiquetas privadas e públicas estejam presentes para a detecção automática ser bem-sucedida.

Considerações

- O AWS Load Balancer Controller criará ALBs e os recursos da AWS de apoio necessários sempre que um recurso de entrada do Kubernetes for criado no cluster com a anotação `kubernetes.io/ingress.class: alb`. O recurso de ingresso configura o ALB para rotear o tráfego HTTP ou HTTPS para diferentes Pods no cluster. Para garantir que os seus objetos de ingresso usem o AWS Load Balancer Controller, adicione a anotação a seguir à sua especificação de ingresso do Kubernetes. Para obter mais informações, consulte Ingress specification (Especificação de ingresso) no GitHub.

```
annotations:
    kubernetes.io/ingress.class: alb
```

> **Note**
>
> Se você estiver balanceando a carga para Pods `IPv6`, adicione a anotação a seguir à sua especificação de ingresso. Você só pode balancear a carga em `IPv6` para destinos IP, e não para destinos de instâncias. Sem essa anotação, o balanceamento de carga ocorrerá em `IPv4`.
>
> ```
> alb.ingress.kubernetes.io/ip-address-type: dualstack
> ```

- O AWS Load Balancer Controller é compatível com os seguintes modos de tráfego:
  - Instance: registra nós dentro do cluster como destinos para o ALB. O tráfego que chega ao ALB é roteado para `NodePort` para o serviço e, depois, enviado por proxy para os Pods. Esse é o modo de tráfego padrão. Também é possível especificá-lo explicitamente com a anotação `alb.ingress.kubernetes.io/target-type: instance`.

    > **Note**
    >
    > O serviço Kubernetes deve especificar o tipo `NodePort` ou “LoadBalancer” para usar esse modo de tráfego.

  - IP: registra Pods como destinos para o ALB. O tráfego que chega ao ALB é roteado diretamente para os Pods do serviço. É necessário especificar a anotação `alb.ingress.kubernetes.io/target-type: ip` para usar esse modo de tráfego. O tipo de destino IP é necessário quando os Pods de destino estão sendo executados no Fargate.
- Para marcar ALBs criados pelo controlador, adicione a seguinte anotação ao controlador: `alb.ingress.kubernetes.io/tags`. Para obter uma lista de todas as anotações disponíveis compatíveis com o AWS Load Balancer Controller, consulte Ingress annotations (Anotações de ingresso) no GitHub.
- Atualizar ou retroceder a versão do controlador de ALB pode introduzir alterações importantes para recursos que dependem dele. Para obter mais informações sobre as alterações que são introduzidas em cada versão, consulte ALB controller release notes (Notas de versão do ALB Controller) no GitHub.

Para compartilhar um balanceador de carga da aplicação entre vários recursos de serviço usando **`IngressGroups`**

Para integrar um ingresso e um grupo, adicione a anotação a seguir a uma especificação do recurso de ingresso do Kubernetes.

```
alb.ingress.kubernetes.io/group.name: my-group
```

O nome do grupo deve:

- Ter no máximo 63 caracteres de comprimento.
- Consistir em letras minúsculas, números, `-` e `.`
- Iniciar e terminar com um número ou letra.

O controlador mescla automaticamente as regras de entrada para todas as entradas no mesmo grupo de entrada. Ele os suporta com um único ALB. A maioria das anotações definidas em uma entrada só se aplica aos caminhos definidos por essa entrada. Por padrão, os recursos de entrada não pertencem a nenhum grupo de entrada.

> **Warning**
>
> Risco potencial de segurança: só especifique um grupo de ingresso para um ingresso quando todos os usuários do Kubernetes com permissão RBAC para criar ou modificar os

recursos de ingresso estiverem dentro da mesma barreira de confiança. Se você adicionar a anotação com um nome de grupo, talvez outros usuários do Kubernetes possam criar ou modificar os ingressos para pertencerem ao mesmo grupo de ingresso. Fazer isso pode causar comportamento indesejável, como substituir regras existentes por regras de prioridade mais alta.

Você pode adicionar um número de ordem para o seu recurso de entrada.

```
alb.ingress.kubernetes.io/group.order: '10'
```

O número pode ser de 1 a 1000. O número mais baixo para todas as entradas no mesmo grupo de entrada será avaliado primeiro. Todas as entradas sem essa anotação são avaliadas com um valor zero. As regras duplicadas com um número mais alto podem substituir as regras com um número mais baixo. Por padrão, a ordem de regra entre as entradas no mesmo grupo de entrada é determinada por ordem léxica com base no namespace e no nome.

Important

Certifique-se de que cada entrada no mesmo grupo de entrada tenha um número de prioridade exclusivo. Não é possível ter números de ordens duplicados nas entradas.

(Opcional) Implantar uma aplicação de exemplo

Pré-requisitos

- Pelo menos uma sub-rede pública ou privada na VPC do cluster.
- Ter o AWS Load Balancer Controller implantado no seu cluster. Para ter mais informações, consulte O que é o AWS Load Balancer Controller?. Recomendamos a versão `2.7.2` ou posterior.

Para implantar uma aplicação de exemplo

É possível executar a aplicação de amostra em um cluster que tenha nós do Amazon EC2, Pods do Fargate ou ambos.

1. Se não estiver implantando no Fargate, ignore esta etapa. Se estiver implantando no Fargate, crie um perfil do Fargate. Você pode criar o perfil executando o comando a seguir ou no AWS

Management Console usando os mesmos valores para `name` e `namespace` que estão no comando. Substitua *`example values`* pelos seus próprios valores.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name alb-sample-app \
    --namespace game-2048
```

2. Implante o jogo 2048 como uma aplicação de exemplo para verificar se o AWS Load Balancer Controller cria um AWS ALB como resultado do objeto de entrada. Realize as etapas para o tipo de sub-rede em que está implantando.

   a. Se você estiver implantando em Pods de um cluster criado com a família `IPv6`, pule para a próxima etapa.

      - Public

        ```
        kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
        ```

      - Private

        1. Faça download do manifesto.

           ```
           curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
           ```

        2. Edite o arquivo e encontre a linha que diz `alb.ingress.kubernetes.io/scheme: internet-facing`.
        3. Altere *`internet-facing`* para **`internal`** e salve o arquivo.
        4. Aplique o manifesto ao cluster.

           ```
           kubectl apply -f 2048_full.yaml
           ```

   b. Se você estiver implantando em Pods de um cluster criado com a família `IPv6`, conclua as etapas a seguir.

      1. Faça download do manifesto.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
```

2. Abra o arquivo em um editor e adicione a linha a seguir às anotações nas especificações de entrada.

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

3. Se você estiver balanceando a carga para Pods internos, em vez de Pods que fazem interface com a Internet, altere a linha que diz `alb.ingress.kubernetes.io/scheme: internet-facing` para `alb.ingress.kubernetes.io/scheme: internal`
4. Salve o arquivo.
5. Aplique o manifesto ao cluster.

```
kubectl apply -f 2048_full.yaml
```

3. Após alguns minutos, verifique se o recurso de entrada foi criado com o comando a seguir.

```
kubectl get ingress/ingress-2048 -n game-2048
```

Veja um exemplo de saída abaixo.

```
NAME           CLASS    HOSTS   ADDRESS
                       PORTS   AGE
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-code.elb.amazonaws.com   80      2m32s
```

> **Note**
>
> Se você criou o balanceador de carga em uma sub-rede privada, o valor de `ADDRESS` na saída anterior é prefaciado com `internal-`.
> Se a sua entrada não for criada após alguns minutos com êxito, execute o comando a seguir para exibir os logs do AWS Load Balancer Controller. Os logs podem conter mensagens de erro que podem ajudar a diagnosticar problemas na sua implantação.

```
kubectl logs -f -n kube-system -l app.kubernetes.io/instance=aws-load-
balancer-controller
```

4. Se implantou em uma sub-rede pública, abra um navegador e acesse o URL `ADDRESS` do resultado do comando anterior para ver a aplicação de exemplo. Se não vir nada, atualize o navegador e tente novamente. Se você implantou em uma sub-rede privada, precisará visualizar a página em um dispositivo dentro da VPC, como um bastion host. Para obter mais informações, consulte Bastion hosts Linux na AWS.
5. Ao terminar de fazer os testes com a aplicação de exemplo, exclua-a executando um dos comandos a seguir.
   - Se você aplicou o manifesto, em vez de aplicar uma cópia que baixou, use o comando a seguir.

     ```
     kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-
     balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
     ```

   - Se você baixou e editou o manifesto, use o comando a seguir.

     ```
     kubectl delete -f 2048_full.yaml
     ```

Restringir endereços IP que podem ser atribuídos a serviços.

Os serviços do Kubernetes podem ser acessados de um cluster por meio de:

- um endereço IP de cluster atribuído automaticamente pelo Kubernetes
- Qualquer endereço IP que você especificar para a propriedade `externalIPs` em uma especificação de serviço. Os endereços IP externos não são gerenciados pelo Kubernetes e são da responsabilidade do administrador do cluster. Endereços IP externos especificados com `externalIPs` são diferentes do endereço IP externo atribuído a um serviço do tipo `LoadBalancer` por um provedor de nuvem.

Para saber mais sobre os serviços do Kubernetes, consulte Services (Serviços) na documentação do Kubernetes. Você pode restringir os endereços IP que podem ser especificados para `externalIPs`em uma especificação de serviço.

Para restringir os endereços IP que podem ser especificados para **`externalIPs`** em uma especificação de serviço

1. Implante `cert-manager` para gerenciar certificados do webhook. Para obter mais informações, consulte a documentação do `cert-manager`.

   ```
   kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert-manager.yaml
   ```

2. Verifique se `cert-manager` Pods estão em execução.

   ```
   kubectl get pods -n cert-manager
   ```

   Veja um exemplo de saída abaixo.

   ```
   NAME                                       READY   STATUS    RESTARTS   AGE
   cert-manager-58c8844bb8-nlx7q              1/1     Running   0          15s
   cert-manager-cainjector-745768f6ff-696h5   1/1     Running   0          15s
   cert-manager-webhook-67cc76975b-4v4nk      1/1     Running   0          14s
   ```

3. Analise os serviços existentes para garantir que nenhum deles tenha endereços IP externos atribuídos a eles que não estejam contidos no bloco CIDR, ao qual você deseja limitar os endereços.

   ```
   kubectl get services -A
   ```

   Veja um exemplo de saída abaixo.

   ```
   NAMESPACE                   NAME                                      TYPE
      CLUSTER-IP       EXTERNAL-IP     PORT(S)         AGE
   cert-manager                cert-manager                              ClusterIP
      10.100.102.137   <none>          9402/TCP        20m
   cert-manager                cert-manager-webhook                      ClusterIP
      10.100.6.136     <none>          443/TCP         20m
   default                     kubernetes                                ClusterIP
      10.100.0.1       <none>          443/TCP         2d1h
   externalip-validation-system   externalip-validation-webhook-service   ClusterIP
      10.100.234.179   <none>          443/TCP         16s
   kube-system                 kube-dns                                  ClusterIP
      10.100.0.10      <none>          53/UDP,53/TCP   2d1h
   ```

```
my-namespace                   my-service                                  ClusterIP
   10.100.128.10    192.168.1.1      80/TCP          149m
```

Se qualquer um dos valores for endereços IP que não estiverem dentro do bloco ao qual você deseja restringir o acesso, você precisará alterar os endereços para que fiquem dentro do bloco e reimplantar os serviços. Por exemplo, o serviço `my-service` na saída anterior tem um endereço IP externo atribuído a ele que não está dentro do exemplo de bloco CIDR na etapa 5.

4. Faça download do manifesto de webhook IP externo. Você também pode visualizar o source code for the webhook (código-fonte para webhook) no GitHub.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
```

5. Especifique blocos CIDR. Abra o arquivo baixado em seu editor e remova o # no início das linhas a seguir.

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

Substitua `10.0.0.0/8` pelo seu próprio bloco CIDR. Especifique quantos blocos quiser. Se especificar vários blocos, adicione uma vírgula entre eles.

6. Se o cluster não estiver na Região da AWS `us-west-2`, substitua `us-west-2`, `602401143452` e `amazonaws.com` no arquivo pelos seguintes comandos: Antes de executar os comandos, substitua *`region-code`* e *`111122223333`* pelo valor da Região da AWS da lista em Visualizar registros de imagens de contêineres da Amazon para complementos do Amazon EKS.

```
sed -i.bak -e 's|602401143452|111122223333|' externalip-webhook.yaml
sed -i.bak -e 's|us-west-2|region-code|' externalip-webhook.yaml
sed -i.bak -e 's|amazonaws.com||' externalip-webhook.yaml
```

7. Aplique o manifesto ao cluster.

```
kubectl apply -f externalip-webhook.yaml
```

Uma tentativa de implantar um serviço no cluster com um endereço IP especificado para `externalIPs`, que não estiver contido nos blocos que você especificou na etapa Especificar blocos CIDR, falhará.

Copiar uma imagem de contêiner de um repositório para outro

Este tópico descreve como extrair uma imagem de contêiner de um repositório ao qual seus nós não têm acesso e enviar a imagem para um repositório ao qual seus nós têm acesso. Você pode enviar a imagem para o Amazon ECR ou para um repositório alternativo ao qual seus nós têm acesso.

Pré-requisitos

- O mecanismo do Docker instalado e configurado no computador. Para obter instruções, consulte Instalar mecanismo do Docker na documentação do Docker.
- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.
- Um endpoint da VPC de interface para o Amazon ECR se você quiser que seus nós extraiam imagens de contêiner ou enviem imagens de contêiner para um repositório privado do Amazon ECR pela rede da Amazon. Para obter mais informações, consulte Criação dos endpoints da VPC para o Amazon ECR no Guia do usuário do Amazon Elastic Container Registry.

Conclua as etapas a seguir para extrair uma imagem de contêiner de um repositório e enviá-la para seu próprio repositório. Nos exemplos fornecidos a seguir neste tópico, a imagem do auxiliar de métricas do Amazon VPC CNI plugin for Kubernetes é extraída. Ao seguir estas etapas, certifique-se de substituir os *`example values`* por seus próprios valores.

Para copiar uma imagem de contêiner de um repositório para outro

1. Se você ainda não tem um repositório do Amazon ECR ou outro repositório, crie um ao qual os nós tenham acesso. O comando a seguir cria um repositório privado do Amazon ECR. Um nome de repositório privado do Amazon ECR deve começar com uma letra. Ele pode conter apenas letras minúsculas, números, hifens (-), sublinhados (_) e barras (/). Para obter mais informações, consulte Criar um repositório privado no Guia do usuário do Amazon Elastic Container Registry.

Você pode substituir *cni-metrics-helper* por qualquer valor que escolher. Como prática recomendada, crie um repositório separado para cada imagem. Recomendamos isso porque as etiquetas de imagem devem ser exclusivas em um repositório. Substitua *region-code* por uma Região da AWS compatível com o Amazon ECR.

```
aws ecr create-repository --region region-code --repository-name cni-metrics-helper
```

2. Determine o registro, o repositório e a etiqueta (opcional) da imagem que seus nós precisam extrair. Essas informações estão no formato `registry/repository[:tag]`.

   Muitos dos tópicos do Amazon EKS sobre a instalação de imagens exigem que você aplique um arquivo manifesto ou instale a imagem usando um chart do Helm. No entanto, antes de aplicar um arquivo manifesto ou instalar um chart do Helm, primeiro visualize o conteúdo do manifesto ou arquivo `values.yaml` do chart. Dessa forma, é possível determinar o registro, o repositório e a etiqueta a serem extraídos.

   Por exemplo, é possível encontrar a seguinte linha no arquivo manifesto para o auxiliar de métricas do Amazon VPC CNI plugin for Kubernetes. O registro é `602401143452.dkr.ecr.us-west-2.amazonaws.com`, que é um registro privado do Amazon ECR. O repositório é `cni-metrics-helper`.

   ```
   image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.12.6"
   ```

   Você poderá ver as seguintes variações para o local de uma imagem:

   - Somente `repository-name:tag`. Nesse caso, `docker.io` geralmente é o registro, mas não especificado, pois o Kubernetes, por padrão, pressupõe que seja um nome de repositório se nenhum Registro for especificado.
   - `repository-name/repository-namespace/repository:tag`. Um namespace de repositório é opcional, mas às vezes é especificado pelo proprietário do repositório para categorizar imagens. Por exemplo, todas as imagens do Amazon EC2 na Galeria Pública do Amazon ECR usam o namespace `aws-ec2`.

   Antes de instalar uma imagem com Helm, veja o arquivo `values.yaml` do Helm para determinar a localização da imagem. Por exemplo, o arquivo `values.yaml` para o auxiliar de métricas do Amazon VPC CNI plugin for Kubernetes inclui as linhas a seguir.

```
image:
  region: us-west-2
  tag: v1.12.6
  account: "602401143452"
  domain: "amazonaws.com"
```

3. Extraia a imagem do contêiner especificada no arquivo manifesto.

   a. Se você estiver extraindo de um registro público, como a Galeria pública do Amazon ECR, é possível pular para a próxima subetapa, porque a autenticação não é necessária. Nesse exemplo, você se autentica em um registro privado do Amazon ECR que contém o repositório da imagem do auxiliar de métricas CNI. O Amazon EKS mantém a imagem em cada registro listado em Visualizar registros de imagens de contêineres da Amazon para complementos do Amazon EKS. Você pode autenticar em qualquer um dos registros substituindo *602401143452* e *region-code* pelas informações de um registro diferente. Existe um registro separado para cada Região da AWS em que o Amazon EKS tem suporte.

   ```
   aws ecr get-login-password --region region-code | docker login --username AWS --password-stdin 602401143452.dkr.ecr.region-code.amazonaws.com
   ```

   b. Extraia a imagem. Nesse exemplo, você extrai do registro no qual se autenticou na subetapa anterior. Substitua *602401143452* e *region-code* pelas informações fornecidas na subetapa anterior.

   ```
   docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
   ```

4. Etiquete a imagem que você extraiu com seu registro, repositório e etiqueta. O exemplo a seguir pressupõe que você extraiu a imagem do arquivo manifesto e a enviará para o repositório privado do Amazon ECR criado na primeira etapa. Substitua *111122223333* pelo ID da sua conta. Substitua *region-code* pela Região da AWS na qual você criou o repositório privado do Amazon ECR.

   ```
   docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
   ```

5. Faça a autenticação em seu registro. Neste exemplo, você autentica no registro privado do Amazon ECR criado na primeira etapa. Para obter mais informações, consulte Registry

authentication (Autenticação de registro) no Guia do usuário do Amazon Elastic Container Registry.

```
aws ecr get-login-password --region region-code | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com
```

6. Envie a imagem ao seu repositório. Neste exemplo, você envia a imagem ao registro privado do Amazon ECR criado na primeira etapa. Para obter mais informações, consulte Envio de uma imagem do Docker no Guia do usuário do Amazon Elastic Container Registry.

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-
helper:v1.12.6
```

7. Atualize o arquivo manifesto usado para determinar a imagem em uma etapa anterior com o `registry/repository:tag` da imagem que você enviou. Se você estiver instalando com um chart do Helm, muitas vezes há uma opção para especificar o `registry/repository:tag`. Ao instalar o chart, especifique o `registry/repository:tag` da imagem que você enviou ao seu repositório.

Visualizar registros de imagens de contêineres da Amazon para complementos do Amazon EKS

Quando você implanta complementos do AWS Amazon EKS no cluster, os nós extraem as imagens de contêiner necessárias do registro especificado no mecanismo de instalação do complemento, como um manifesto de instalação ou um arquivo `values.yaml` do Helm. As imagens são extraídas de um repositório privado do Amazon ECR no Amazon EKS. O Amazon EKS replica as imagens para um repositório em cada Região da AWS compatível com o Amazon EKS. Seus nós podem extrair a imagem do contêiner pela Internet de qualquer um dos registros a seguir. Como alternativa, seus nós podem extrair a imagem pela rede da Amazon se você criou um endpoint da VPC de interface para o Amazon ECR (AWS PrivateLink) na sua VPC. Os registros exigem autenticação com uma conta do AWS IAM. Seus nós são autenticados usando a função do IAM de nó do Amazon EKS, que possui as permissões na política do IAM gerenciada AmazonEC2ContainerRegistryReadOnly associada a ela.

Região da AWS	Registro
af-south-1	877085696533.dkr.ecr.af-south-1.amazonaws.com
ap-east-1	800184023465.dkr.ecr.ap-east-1.amazonaws.com
ap-northeast-1	602401143452.dkr.ecr.ap-northeast-1.amazonaws.com
ap-northeast-2	602401143452.dkr.ecr.ap-northeast-2.amazonaws.com
ap-northeast-3	602401143452.dkr.ecr.ap-northeast-3.amazonaws.com
ap-south-1	602401143452.dkr.ecr.ap-south-1.amazonaws.com/
ap-south-2	900889452093.dkr.ecr.ap-south-2.amazonaws.com
ap-southeast-1	602401143452.dkr.ecr.ap-southeast-1.amazonaws.com
ap-southeast-2	602401143452.dkr.ecr.ap-southeast-2.amazonaws.com
ap-southeast-3	296578399912.dkr.ecr.ap-southeast-3.amazonaws.com
ap-southeast-4	491585149902.dkr.ecr.ap-southeast-4.amazonaws.com
ca-central-1	602401143452.dkr.ecr.ca-central-1.amazonaws.com

Região da AWS	Registro
ca-west-1	761377655185.dkr.ecr.ca-west-1.amazonaws.com
cn-north-1	918309763551.dkr.ecr.cn-north-1.amazonaws.com.cn
cn-northwest-1	961992271922.dkr.ecr.cn-northwest-1.amazonaws.com.cn
eu-central-1	602401143452.dkr.ecr.eu-central-1.amazonaws.com
eu-central-2	900612956339.dkr.ecr.eu-central-2.amazonaws.com
eu-north-1	602401143452.dkr.ecr.eu-north-1.amazonaws.com
eu-south-1	590381155156.dkr.ecr.eu-south-1.amazonaws.com
eu-south-2	455263428931.dkr.ecr.eu-south-2.amazonaws.com
eu-west-1	602401143452.dkr.ecr.eu-west-1.amazonaws.com
eu-west-2	602401143452.dkr.ecr.eu-west-2.amazonaws.com/
eu-west-3	602401143452.dkr.ecr.eu-west-3.amazonaws.com/
il-central-1	066635153087.dkr.ecr.il-central-1.amazonaws.com
me-south-1	558608220178.dkr.ecr.me-south-1.amazonaws.com/

Região da AWS	Registro
me-central-1	759879836304.dkr.ecr.me-central-1.amazonaws.com
sa-east-1	602401143452.dkr.ecr.sa-east-1.amazonaws.com/
us-east-1	602401143452.dkr.ecr.us-east-1.amazonaws.com/
us-east-2	602401143452.dkr.ecr.us-east-2.amazonaws.com/
us-gov-east-1	151742754352.dkr.ecr.us-gov-east-1.amazonaws.com/
us-gov-west-1	013241004608.dkr.ecr.us-gov-west-1.amazonaws.com/
us-west-1	602401143452.dkr.ecr.us-west-1.amazonaws.com
us-west-2	602401143452.dkr.ecr.us-west-2.amazonaws.com/

Complementos do Amazon EKS

Um complemento é um software que fornece recursos operacionais de suporte a aplicações do Kubernetes, mas que não é específico da aplicação. Isso inclui software como agentes de capacidade de observação ou drivers do Kubernetes que permitem que o cluster interaja com recursos da AWS para conexão de rede, computação e armazenamento. O software complementar é normalmente criado e mantido pela comunidade do Kubernetes, provedores de nuvem como a AWS ou fornecedores terceirizados. O Amazon EKS instala automaticamente complementos autogerenciados, como o Amazon VPC CNI plugin for Kubernetes, o `kube-proxy` e o CoreDNS, para cada cluster. Você pode alterar a configuração padrão dos complementos e atualizá-los quando desejar.

Os complementos do Amazon EKS fornecem instalação e gerenciamento de um conjunto de complementos selecionados especialmente para os clusters do Amazon EKS. Todos os complementos do Amazon EKS incluem os patches de segurança mais recentes e as correções de bugs, e são validados pela AWS para funcionar com o Amazon EKS. Os complementos do Amazon EKS permitem que você garanta consistentemente que os clusters do Amazon EKS estejam seguros e estáveis, além de reduzirem a quantidade de trabalho que você precisa fazer para instalar, configurar e atualizar complementos. Se um complemento autogerenciado, como o `kube-proxy`, já estiver em execução em seu cluster e estiver disponível como um complemento do Amazon EKS, você pode instalar o complemento `kube-proxy` do Amazon EKS para começar a se beneficiar dos recursos dos complementos do Amazon EKS.

Você pode atualizar campos específicos de configuração gerenciados pelo Amazon EKS por meio da API do Amazon EKS. Você também pode modificar os campos de configuração não gerenciados pelo Amazon EKS diretamente no cluster do Kubernetes, assim que o complemento é iniciado. Isso inclui a definição dos campos de configuração específicos para um complemento, quando aplicável. Essas alterações não serão substituídas pelo Amazon EKS depois de feitas. Isso é possível usando o recurso de aplicação do lado do servidor do Kubernetes. Para obter mais informações, consulte Determinar os campos que podem ser personalizados para os complementos do Amazon EKS.

É possível usar os complementos do Amazon EKS com qualquer tipo de nó do Amazon EKS.

Você pode adicionar, atualizar ou excluir complementos do Amazon EKS usando a API do Amazon EKS, o AWS Management Console, a AWS CLI e o `eksctl`. Você também pode criar complementos do Amazon EKS usando o AWS CloudFormation.

Considerações

Considere o seguinte ao usar complementos do Amazon EKS:

- Para configurar complementos para o cluster, a entidade principal do IAM deve ter permissões do IAM para trabalhar com complementos. Para obter mais informações, consulte as ações com `Addon` no nome, em Ações definidas pelo Amazon Elastic Kubernetes Service.
- Os complementos do Amazon EKS são executados nos nós que você provisionar ou configurar para o cluster. Os tipos de nó incluem instâncias do Amazon EC2 e do Fargate.
- Você pode modificar campos que não são gerenciados pelo Amazon EKS para personalizar a instalação de um complemento do Amazon EKS. Para obter mais informações, consulte Determinar os campos que podem ser personalizados para os complementos do Amazon EKS.

- Se você criar um cluster com o AWS Management Console, o `kube-proxy` do Amazon EKS, o Amazon VPC CNI plugin for Kubernetes e os complementos CoreDNS do Amazon EKS serão adicionados automaticamente ao cluster. Se você usar `eksctl` para criar o cluster com um arquivo `config`, o `eksctl` também poderá criar o cluster com complementos do Amazon EKS. Se você criar o cluster usando `eksctl` sem o arquivo `config`, ou com qualquer outra ferramenta, os complementos `kube-proxy` autogerenciado, Amazon VPC CNI plugin for Kubernetes e CoreDNS serão instalados no lugar dos complementos do Amazon EKS. Você pode gerenciá-los você mesmo ou pode adicionar os complementos do Amazon EKS manualmente após a criação do cluster.
- O `eks:addon-cluster-admin` `ClusterRoleBinding` vincula o `cluster-admin``ClusterRole` à identidade do `eks:addon-manager` Kubernetes. O perfil tem as permissões necessárias para que a identidade do `eks:addon-manager` crie namespaces do Kubernetes e instale complementos nos namespaces. Se o `ClusterRoleBinding` `eks:addon-cluster-admin` for removido, o cluster do Amazon EKS continuará funcionando, mas o Amazon EKS não poderá mais gerenciar os complementos. Todos os clusters que começam com as seguintes versões da plataforma usam o novo `ClusterRoleBinding`.

Versão da plataforma do Amazon EKS	Versão do Kubernetes
eks.2	1.20.12
eks.1	1.21.14
eks.2	1.22.9
eks.2	1.23.5
eks.3	1.24.3

Complementos do Amazon EKS disponíveis da AWS

Os complementos do Amazon EKS a seguir estão disponíveis para serem criados no seu cluster. Sempre é possível ver a lista mais atual de complementos disponíveis usando `eksctl`, o AWS

Management Console ou a AWS CLI. Para ver todos os complementos disponíveis ou instalar um complemento, consulte Criar um complemento do Amazon EKS. Se um complemento precisar de permissões do IAM, você deve ter um provedor do IAM OpenID Connect (OIDC) para seu cluster. Para determinar se você já tem um ou se precisa criar um, consulte Criar um provedor OIDC do IAM para o cluster. Você pode atualizar ou excluir um complemento após instalá-lo.

É possível usar qualquer um dos complementos do Amazon EKS a seguir.

Descrição	Saiba mais
Fornecer uma rede VPC nativa para seu cluster	Amazon VPC CNI plugin for Kubernetes
Um servidor DNS flexível e extensível que pode servir como o DNS de cluster do Kubernetes	CoreDNS
Manter as regras de rede em cada nó do Amazon EC2	`Kube-proxy`
Fornecer armazenamento do Amazon EBS para seu cluster	Driver da CSI do Amazon EBS
Fornecer armazenamento do Amazon EFS para seu cluster	Driver da CSI do Amazon EFS
Fornecer armazenamento do Amazon S3 para seu cluster	Mountpoint para driver da CSI do Amazon S3
Habilite o uso da funcional idade de captura de snapshots em drivers da CSI compatíve is, como o driver da CSI do Amazon EBS	Controlador de snapshots da CSI
Distribuição do projeto OpenTelemetry segura, pronta	AWS Distro para OpenTelem etry

Descrição	Saiba mais	
para produção e com suporte da AWS		
Serviço de monitoramento de segurança que analisa e processa fontes de dados básicas, incluindo eventos de gerenciamento do AWS CloudTrail e logs de fluxo da Amazon VPC. O Amazon GuardDuty também processa recursos como logs de auditoria do Kubernetes e monitoramento de runtime	agente do Amazon GuardDuty	
Serviço de monitoramento e observabilidade fornecido pela AWS. Esse complemento instala o CloudWatch Agent e habilita o CloudWatch Application Signals e o CloudWatch Container Insights com observabilidade aprimorada para o Amazon EKS	Agente do Amazon CloudWatch Observability	
Capacidade de gerenciar credenciais para suas aplicações de forma semelhante a como os perfis de instância do EC2 fornecem credenciais para instâncias do EC2	EKS Pod Identity Agent	

Amazon VPC CNI plugin for Kubernetes

O complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS é um plug-in de Container Network Interface (CNI) do Kubernetes que fornece rede VPC nativa para seu cluster. O tipo autogerenciado ou gerenciado desse complemento é instalado por padrão em cada nó do Amazon EC2. Para obter mais informações, consulte Plug-in de Container Network Interface (CNI) do Kubernetes.

O nome do complemento do Amazon EKS é `vpc-cni`.

Permissões obrigatórias do IAM

Este complemento usa o recurso de perfis do IAM para contas de serviço do Amazon EKS. Se seu cluster usar a família `IPv4`, é necessário ter as permissões na AmazonEKS_CNI_Policy. Se o seu cluster utilizar a família `IPv6`, é necessário criar uma política do IAM com as permissões no modo IPv6. Você pode criar um perfil do IAM, anexar uma das políticas a ele e anotar a conta de serviço do Kubernetes usada pelo complemento com o comando a seguir.

Substitua *`my-cluster`* pelo nome do seu cluster e substitua *`AmazonEKSVPCCNIRole`* pelo nome desejado para o seu perfil. Se seu cluster usar a família `IPv6`, substitua *`AmazonEKS_CNI_Policy`* pelo nome da política que você criou. Esse comando requer que você tenha o `eksctl` instalado no dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil, anexar a política a ele e anotar na conta de serviço do Kubernetes, consulte Atribuir perfis do IAM às contas de serviço do Kubernetes.

```
eksctl create iamserviceaccount --name aws-node --namespace kube-system --cluster my-
cluster --role-name AmazonEKSVPCCNIRole \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy --
approve
```

Informações sobre a atualização

Você só pode atualizar uma versão secundária de cada vez. Por exemplo, se a versão atual for `1.28.`*`x`*`-eksbuild.`*`y`* e você quiser atualizar para `1.30.`*`x`*`-eksbuild.`*`y`*, será necessário atualizar sua versão atual para `1.29.`*`x`*`-eksbuild.`*`y`* e depois atualizá-la novamente para a versão `1.30.`*`x`*`-eksbuild.`*`y`*. Para obter mais informações sobre como atualizar o complemento, consulte Atualizar o complemento do Amazon EKS.

CoreDNS

O complemento CoreDNS do Amazon EKS é um servidor DNS flexível e extensível que pode servir como DNS de cluster do Kubernetes. Por padrão, o tipo autogerenciado ou gerenciado desse complemento foi instalado quando você criou seu cluster. Quando você executa um cluster do Amazon EKS com pelo menos um nó, duas réplicas da imagem do CoreDNS são implantadas por padrão, independentemente do número de nós implantados em seu cluster. Os CoreDNS Pods fornecem resolução de nomes para todos os Pods no cluster. É possível implantar os Pods do CoreDNS em nós do Fargate se o seu cluster incluir um Defina quais Pods usarão o AWS Fargate quando em execução com um namespace correspondente ao namespace para a `deployment` do CoreDNS.

O nome do complemento do Amazon EKS é `coredns`.

Permissões obrigatórias do IAM

Este complemento não exige nenhuma permissão.

Mais informações

Para saber mais sobre o CoreDNS, consulte Usar o CoreDNS para descoberta de serviços e Personalizar o serviço de DNS na documentação do Kubernetes.

`Kube-proxy`

O complemento `Kube-proxy` do Amazon EKS mantém as regras de rede em cada nó do Amazon EC2. Esse componente viabiliza a comunicação de rede com seus Pods. O tipo autogerenciado ou gerenciado desse complemento é instalado por padrão em cada nó do Amazon EC2 em seu cluster.

O nome do complemento do Amazon EKS é `kube-proxy`.

Permissões obrigatórias do IAM

Este complemento não exige nenhuma permissão.

Informações sobre a atualização

Antes de atualizar sua versão atual, considere os seguintes requisitos:

- O `Kube-proxy` em um cluster do Amazon EKS tem a mesma política de compatibilidade e distorção que o Kubernetes.

Mais informações

Para saber mais sobre o `kube-proxy`, consulte `kube-proxy` na documentação do Kubernetes.

Driver da CSI do Amazon EBS

O complemento de driver da CSI do Amazon EBS do Amazon EKS é um plug-in de Container Storage Interface (CSI) do Kubernetes que fornece armazenamento do Amazon EBS para o cluster.

O nome do complemento do Amazon EKS é `aws-ebs-csi-driver`.

Permissões obrigatórias do IAM

Este complemento utiliza o recurso de perfis do IAM para contas de serviço do Amazon EKS. As permissões na `AmazonEBSCSIDriverPolicy`AWS política gerenciada são obrigatórias. Você pode criar um perfil do IAM e anexar a política gerenciada a ela com o seguinte comando. Substitua *`my-cluster`* pelo nome do seu cluster e substitua *`AmazonEKS_EBS_CSI_DriverRole`* pelo nome desejado para o seu perfil. Esse comando requer que você tenha o `eksctl` instalado no dispositivo. Se você precisar usar uma ferramenta diferente ou precisar usar uma chave KMS personalizada para criptografia, consulte Etapa 1: Criar uma função do IAM.

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
    --approve
```

Mais informações

Para saber mais a respeito do complemento, consulte Armazene volumes do Kubernetes com o Amazon EBS.

Driver da CSI do Amazon EFS

O complemento de driver da CSI do Amazon EFS do Amazon EKS é um plug-in de Container Storage Interface (CSI) do Kubernetes que fornece armazenamento do Amazon EFS para o cluster.

O nome do complemento do Amazon EKS é `aws-efs-csi-driver`.

Permissões obrigatórias do IAM

Permissões necessárias do IAM: este complemento utiliza perfis do IAM para recursos de contas de serviço do Amazon EKS. As permissões na `AmazonEFSCSIDriverPolicy`AWS política gerenciada são obrigatórias. Você pode criar um perfil do IAM e anexar a política gerenciada a ele com o seguinte comando. Substitua *`my-cluster`* pelo nome do seu cluster e substitua *`AmazonEKS_EFS_CSI_DriverRole`* pelo nome desejado para o seu perfil. Esse comando requer que o `eksctl` esteja instalado no seu dispositivo. Se precisar usar outra ferramenta, consulte Etapa 1: Criar uma função do IAM.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
    --approve
TRUST_POLICY=$(aws iam get-role --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

Mais informações

Para saber mais a respeito do complemento, consulte Armazenar um sistema de arquivos elástico com o Amazon EFS.

Mountpoint para driver da CSI do Amazon S3

O complemento de driver da CSI do Mountpoint para Amazon S3 do Amazon EKS é um plug-in de Container Storage Interface (CSI) do Kubernetes que fornece armazenamento do Amazon S3 para o cluster.

O nome do complemento do Amazon EKS é `aws-mountpoint-s3-csi-driver`.

Permissões obrigatórias do IAM

Este complemento usa o recurso de perfis do IAM para contas de serviço do Amazon EKS. O perfil do IAM criado exigirá uma política que conceda acesso ao S3. Siga as recomendações de permissões do IAM do Mountpoint ao criar a política. Como alternativa, é possível usar a política AmazonS3FullAccess gerenciada pela AWS, mas essa política gerenciada concede mais permissões do que as necessárias para o Mountpoint.

Você pode criar um perfil do IAM e anexar a política gerenciada a ele com os comandos a seguir. Substitua *my-cluster* pelo nome do seu cluster, *region-code* pelo código da Região da AWS correta, *AmazonEKS_S3_CSI_DriverRole* pelo nome do seu perfil e *AmazonEKS_S3_CSI_DriverRole_ARN* pelo ARN do perfil. Esse comando requer que o `eksctl` esteja instalado no seu dispositivo. Para obter instruções sobre como usar o console do IAM ou AWS CLI, consulte Criar um perfil do IAM.

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

Mais informações

Para saber mais a respeito do complemento, consulte Armazenar dados com a interface web do Amazon S3.

Controlador de snapshots da CSI

O controlador de snapshots da Container Storage Interface (CSI) permite utilizar a funcionalidade de captura de snapshots em drivers da CSI compatíveis, como o driver da CSI do Amazon EBS.

O nome do complemento do Amazon EKS é `snapshot-controller`.

Permissões obrigatórias do IAM

Este complemento não exige nenhuma permissão.

Mais informações

Para saber mais a respeito do complemento, consulte Habilitar a funcionalidade de snapshot para volumes CSI.

AWS Distro para OpenTelemetry

O complemento AWS Distro for OpenTelemetry do Amazon EKS uma distribuição segura, pronta para produção e com suporte da AWS do projeto OpenTelemetry. Para obter mais informações, consulte AWS Distro for OpenTelemetry no GitHub.

O nome do complemento do Amazon EKS é `adot`.

Permissões obrigatórias do IAM

Este complemento só exigirá permissões do IAM se você estiver usando um dos recursos personalizados pré-configurados que podem ser escolhidos via configuração avançada.

Mais informações

Para obter informações adicionais, consulte Conceitos básicos do AWS Distro para OpenTelemetry usando complementos do EKS na documentação do AWS Distro para OpenTelemetry.

O ADOT exige que o `cert-manager` seja implantado no cluster como um pré-requisito, caso contrário, esse complemento não funcionará se implantado diretamente usando a propriedade `cluster_addons` do Amazon EKS Terraform. Para obter mais requisitos, consulte Requisitos para começar a usar o AWS Distro para OpenTelemetry usando complementos do EKS na documentação do AWS Distro para OpenTelemetry.

agente do Amazon GuardDuty

O complemento de agente do Amazon GuardDuty do Amazon EKS é um serviço de monitoramento de segurança que analisa e processa fontes de dados básicas, incluindo eventos de gerenciamento do AWS CloudTrail e logs de fluxo da Amazon VPC. O Amazon GuardDuty também processa recursos, como logs de auditoria do Kubernetes e monitoramentos de runtime.

O nome do complemento do Amazon EKS é `aws-guardduty-agent`.

Permissões obrigatórias do IAM

Este complemento não exige nenhuma permissão.

Mais informações

Para obter mais informações, consulte Monitoramento do runtime para clusters do Amazon EKS no Amazon GuardDuty.

- Para detectar possíveis ameaças de segurança em seus clusters do Amazon EKS, habilite o monitoramento de runtime do Amazon GuardDuty e implante o agente de segurança do GuardDuty em seus clusters do Amazon EKS.

Agente do Amazon CloudWatch Observability

O complemento de agente do Amazon CloudWatch Observability do Amazon EKS aprimora o serviço de monitoramento e observabilidade fornecido pela AWS. Esse complemento instala o CloudWatch Agent e habilita o CloudWatch Application Signals e o CloudWatch Container Insights com observabilidade aprimorada para o Amazon EKS. Para obter mais informações, consulte Agente do Amazon CloudWatch.

O nome do complemento do Amazon EKS é `amazon-cloudwatch-observability`.

Permissões obrigatórias do IAM

Este complemento usa o recurso de perfis do IAM para contas de serviço do Amazon EKS. As permissões nas políticas gerenciadas pela AWS AWSXrayWriteOnlyAccess e CloudWatchAgentServerPolicy são necessárias. Você pode criar um perfil do IAM, anexar as políticas gerenciadas a ele e anotar a conta de serviço do Kubernetes usada pelo complemento com o comando a seguir. Substitua *my-cluster* pelo nome do seu cluster e substitua *AmazonEKS_Observability_role* pelo nome desejado para o seu perfil. Esse comando requer que você tenha o `eksctl` instalado no dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil, anexar a política a ele e anotar na conta de serviço do Kubernetes, consulte Atribuir perfis do IAM às contas de serviço do Kubernetes.

```
eksctl create iamserviceaccount \
    --name cloudwatch-agent \
    --namespace amazon-cloudwatch \
    --cluster my-cluster \
    --role-name AmazonEKS_Observability_Role \
    --role-only \
```

```
    --attach-policy-arn arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
    --approve
```

Mais informações

Para saber mais, consulte Instalar o agente do CloudWatch.

EKS Pod Identity Agent

O complemento Amazon EKS Pod Identity Agent do Amazon EKS permite gerenciar credenciais para suas aplicações de forma semelhante a como os perfis de instância do EC2 fornecem credenciais para instâncias do EC2.

O nome do complemento do Amazon EKS é `eks-pod-identity-agent`.

Permissões obrigatórias do IAM

Este complemento usa permissões do Perfil do IAM em nós do Amazon EKS.

Informações sobre a atualização

Você só pode atualizar uma versão secundária de cada vez. Por exemplo, se a versão atual for `1.28.`*`x`*`-eksbuild.`*`y`* e você quiser atualizar para `1.30.`*`x`*`-eksbuild.`*`y`*, será necessário atualizar sua versão atual para `1.29.`*`x`*`-eksbuild.`*`y`* e depois atualizá-la novamente para a versão `1.30.`*`x`*`-eksbuild.`*`y`*. Para obter mais informações sobre como atualizar o complemento, consulte Atualizar o complemento do Amazon EKS.

Complementos do Amazon EKS de fornecedores independentes de software

Além da lista anterior de complementos do Amazon EKS, você também pode adicionar uma ampla seleção de complementos de software operacional do Amazon EKS de fornecedores de software independentes. Escolha um complemento para saber mais sobre ele e sobre seus requisitos de instalação.

Encontre, adquira e implante complementos do AWS Marketplace no Amazon EKS (YouTube).

Accuknox

O nome do complemento é `accuknox_kubearmor` e o namespace é `kubearmor`. A Accuknox publica o complemento.

Para obter informações sobre o complemento, consulte Introdução ao KubeArmor na documentação da KubeArmor.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Akuity

O nome do complemento é `akuity_agent` e o namespace é `akuity`. A Akuity publica o complemento.

Para obter informações sobre como usar o complemento, consulte Instalar o Akuity Agent no Amazon EKS com o complemento Akuity do EKS na documentação do Akuity Platform.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Calyptia

O nome do complemento é `calyptia_fluent-bit` e o namespace é `calytia-fluentbit`. A Calyptia publica o complemento.

Para obter informações sobre o complemento, consulte Conceitos básicos do Calyptia Core Agent no site de documentação do Calyptia.

Nome da conta de serviço

O nome da conta de serviço é `clyptia-fluentbit`.

Política do IAM gerenciada da AWS

Este complemento usa a política gerenciada `AWSMarketplaceMeteringRegisterUsage`. Para obter mais informações sobre essa política, consulte AWSMarketplaceMeteringRegisterUsage no Guia de referência de políticas gerenciadas pela AWS.

Comando para criar o perfil do IAM necessário

O comando a seguir exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte Criar um provedor OIDC do IAM para o cluster. Substitua *`my-cluster`* pelo nome do seu cluster e substitua *`my-calyptia-role`* pelo nome desejado para o seu perfil. Esse comando requer que você tenha o `eksctl` instalado no dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil e anotar na conta de serviço Kubernetes, consulte Atribuir perfis do IAM às contas de serviço do Kubernetes.

```
eksctl create iamserviceaccount --name service-account-name  --namespace calyptia-
fluentbit --cluster my-cluster --role-name my-calyptia-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/
AWSMarketplaceMeteringRegisterUsage --approve
```

Cisco Observability Collector

O nome do complemento é `cisco_cisco-cloud-observability-collectors` e o namespace é `appdynamics`. A Cisco publica o complemento.

Para obter informações sobre o complemento, consulte Usar os complementos do Cisco Cloud Observability no AWS Marketplace na documentação do Cisco AppDynamics.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Cisco Observability Operator

O nome do complemento é `cisco_cisco-cloud-observability-operators` e o namespace é `appdynamics`. A Cisco publica o complemento.

Para obter informações sobre o complemento, consulte Usar os complementos do Cisco Cloud Observability no AWS Marketplace na documentação do Cisco AppDynamics.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

CLOUDSOFT

O nome do complemento é `cloudsoft_cloudsoft-amp` e o namespace é `cloudsoft-amp`. A CLOUDSOFT publica o complemento.

Para obter informações sobre o complemento, consulte Complemento do Amazon EKS na documentação da CLOUDSOFT.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Cribl

O nome do complemento é `cribl_cribledge` e o namespace é `cribledge`. A Cribl publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento Cribl do Amazon EKS na documentação da Cribl

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Dynatrace

O nome do complemento é `dynatrace_dynatrace-operator` e o namespace é `dynatrace`. A Dynatrace publica o complemento.

Para obter informações sobre o complemento, consulte Monitoramento do Kubernetes na documentação da dynatrace.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Datree

O nome do complemento é `datree_engine-pro` e o namespace é `datree`. A Datree publica o complemento.

Para obter informações sobre o complemento, consulte Integração do Amazon EKS na documentação da Datree.

Nome da conta de serviço

O nome da conta de serviço é datree-webhook-server-awsmp.

Política do IAM gerenciada da AWS

A política gerenciada é AWSLicenseManagerConsumptionPolicy. Para obter mais informações sobre essa política, consulte AWSLicenseManagerConsumptionPolicy no Guia de referência de políticas gerenciadas pela AWS.

Comando para criar o perfil do IAM necessário

O comando a seguir exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte Criar um provedor OIDC do IAM para o cluster. Substitua *my-cluster* pelo nome do seu cluster e substitua *my-datree-role* pelo nome desejado para o seu perfil. Esse comando requer que você tenha o `eksctl` instalado no dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil e anotar na conta de serviço Kubernetes, consulte Atribuir perfis do IAM às contas de serviço do Kubernetes.

```
eksctl create iamserviceaccount --name datree-webhook-server-awsmp --namespace datree
 --cluster my-cluster --role-name my-datree-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Permissões personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Datadog

O nome do complemento é `datadog_operator` e o namespace é `datadog-agent`. A Datadog publica o complemento.

Para obter informações sobre o complemento, consulte Instalação do Datadog Agent no Amazon EKS com o complemento Datadog Operator na documentação da Datadog.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Groundcover

O nome do complemento é `groundcover_agent` e o namespace é `groundcover`. A groundcover publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento Groundcover do Amazon EKS na documentação da Groundcover.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Grafana Labs

O nome do complemento é `grafana-labs_kubernetes-monitoring` e o namespace é `monitoring`. A Grafana Labs publica o complemento.

Para obter informações sobre o complemento, consulte Configurar o monitoramento do Kubernetes como um complemento com o Amazon EKS na documentação da Grafana Labs.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Guance

- Editora: GUANCE
- Nome: `guance_datakit`
- Namespace: `datakit`
- Nome da conta de serviço: nenhuma conta de serviço é usada com esse complemento.
- Política gerenciada do IAM da AWS: nenhuma política gerenciada é usada com esse complemento.
- Permissões personalizadas do IAM: nenhuma permissão personalizada é usada com esse complemento.
- Instruções de configuração e uso: consulte Usar o complemento do Amazon EKS na documentação da Guance.

HA Proxy

O nome é `haproxy-technologies_kubernetes-ingress-ee` e o namespace é `haproxy-controller`. O HA Proxy publica o complemento.

Para obter informações sobre o complemento, consulte Integração do Amazon EKS na documentação da Datree.

Nome da conta de serviço

O nome da conta de serviço é `customer defined`.

Política do IAM gerenciada da AWS

A política gerenciada é AWSLicenseManagerConsumptionPolicy. Para obter mais informações sobre essa política, consulte AWSLicenseManagerConsumptionPolicy no Guia de referência de políticas gerenciadas pela AWS.

Comando para criar o perfil do IAM necessário

O comando a seguir exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte Criar um provedor OIDC

do IAM para o cluster. Substitua *my-cluster* pelo nome do seu cluster e substitua *my-haproxy-role* pelo nome desejado para o seu perfil. Esse comando requer que você tenha o eksctl instalado no dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil e anotar na conta de serviço Kubernetes, consulte Atribuir perfis do IAM às contas de serviço do Kubernetes.

```
eksctl create iamserviceaccount --name service-account-name  --namespace haproxy-
controller --cluster my-cluster --role-name my-haproxy-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Permissões personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Kpow

O nome do complemento é `factorhouse_kpow` e o namespace é `factorhouse`. A Factorhouse publica o complemento.

Para obter mais informações sobre o complemento, consulte AWS Marketplace LM na documentação do Kpow.

Nome da conta de serviço

O nome da conta de serviço é `kpow`.

Política do IAM gerenciada da AWS

A política gerenciada é AWSLicenseManagerConsumptionPolicy. Para obter mais informações sobre essa política, consulte AWSLicenseManagerConsumptionPolicy no Guia de referência de políticas gerenciadas pela AWS.

Comando para criar o perfil do IAM necessário

O comando a seguir exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte Criar um provedor OIDC do IAM para o cluster. Substitua *my-cluster* pelo nome do seu cluster e substitua *my-kpow-role* pelo nome desejado para o seu perfil. Esse comando requer que você tenha o eksctl instalado no dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil e anotar na conta de serviço Kubernetes, consulte Atribuir perfis do IAM às contas de serviço do Kubernetes.

```
eksctl create iamserviceaccount --name kpow --namespace factorhouse --cluster my-
cluster --role-name my-kpow-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Permissões personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Kubecost

O nome do complemento é `kubecost_kubecost` e o namespace é `kubecost`. A Kubecost publica o complemento.

Para obter mais informações sobre o complemento, consulte Integração de cobrança na Nuvem AWS na documentação do Kubecost.

Se seu cluster for da versão `1.23` ou posterior, você deverá ter o the section called “Amazon EBS” instalado no cluster, do contrário receberá um erro.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Kasten

O nome do complemento é `kasten_k10` e o namespace é `kasten-io`. A Kasten by Veeam publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o K10 na AWS usando o complemento do Amazon EKS na documentação da Kasten.

Se a versão do seu cluster do Amazon EKS for Kubernetes `1.23` ou posterior, o driver de CSI do Amazon EBS deverá estar instalado no cluster com um `StorageClass` padrão.

Nome da conta de serviço

O nome da conta de serviço é `k10-k10`.

Política do IAM gerenciada da AWS

A política gerenciada é AWSLicenseManagerConsumptionPolicy. Para obter mais informações sobre essa política, consulte AWSLicenseManagerConsumptionPolicy no Guia de referência de políticas gerenciadas pela AWS.

Comando para criar o perfil do IAM necessário

O comando a seguir exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte Criar um provedor OIDC do IAM para o cluster. Substitua *my-cluster* pelo nome do seu cluster e substitua *my-kasten-role* pelo nome desejado para o seu perfil. Esse comando requer que você tenha o `eksctl` instalado no dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil e anotar na conta de serviço Kubernetes, consulte Atribuir perfis do IAM às contas de serviço do Kubernetes.

```
eksctl create iamserviceaccount --name k10-k10 --namespace kasten-io --cluster my-
cluster --role-name my-kasten-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Permissões personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Kong

O nome do complemento é `kong_konnect-ri` e o namespace é `kong`. A Kong publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento Kong Gateway do EKS na documentação da Kong.

Se seu cluster for da versão `1.23` ou posterior, você deverá ter o the section called “Amazon EBS” instalado no cluster, do contrário receberá um erro.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

LeakSignal

O nome do complemento é `leaksignal_leakagent` e o namespace é `leakagent`. A LeakSignal publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento LeakAgent na documentação da LeakSignal

Se seu cluster for da versão `1.23` ou posterior, você deverá ter o the section called “Amazon EBS” instalado no cluster, do contrário receberá um erro.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

NetApp

O nome do complemento é `netapp_trident-operator` e o namespace é `trident`. A NetApp publica o complemento.

Para obter informações sobre o complemento, consulte Configurar o complemento Astra Trident do EKS na documentação da NetApp.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

New Relic

O nome do complemento é `new-relic_kubernetes-operator` e o namespace é `newrelic`. A New Relic publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento New Relic para EKS na documentação da New Relic.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Rafay

O nome do complemento é `rafay-systems_rafay-operator` e o namespace é `rafay-system`. A Rafay publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento Rafay do Amazon EKS na documentação da Rafay.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Rad Security

- Editora: RAD SECURITY
- Nome: `rad-security_rad-security`
- Namespace: `ksoc`
- Nome da conta de serviço: nenhuma conta de serviço é usada com esse complemento.
- Política gerenciada do IAM da AWS: nenhuma política gerenciada é usada com esse complemento.
- Permissões personalizadas do IAM: nenhuma permissão personalizada é usada com esse complemento.
- Instruções de configuração e uso: consulte Instalar o Rad por meio do AWS Marketplace na documentação da Rad Security.

SolarWinds

- Editora: SOLARWINDS
- Nome: `solarwinds_swo-k8s-collector-addon`
- Namespace: `solarwinds`
- Nome da conta de serviço: nenhuma conta de serviço é usada com esse complemento.
- Política gerenciada do IAM da AWS: nenhuma política gerenciada é usada com esse complemento.
- Permissões personalizadas do IAM: nenhuma permissão personalizada é usada com esse complemento.
- Instruções de configuração e uso: consulte Monitorar um cluster do Amazon EKS na documentação da SolarWinds.

Solo

O nome do complemento é `solo-io_istio-distro` e o namespace é `istio-system`. A Solo publica o complemento.

Para obter mais informações sobre o complemento, consulte Instalar o Istio na documentação da Solo.io.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Snyk

- Editora: SNYK
- Nome: `snyk_runtime-sensor`
- Namespace: `snyk_runtime-sensor`
- Nome da conta de serviço: nenhuma conta de serviço é usada com esse complemento.
- Política gerenciada do IAM da AWS: nenhuma política gerenciada é usada com esse complemento.
- Permissões personalizadas do IAM: nenhuma permissão personalizada é usada com esse complemento.
- Instruções de configuração e uso: consulte o Sensor de tempo de execução do Snyk nos documentos do usuário da Snyk.

Stormforge

O nome do complemento é `stormforge_optimize-Live` e o namespace é `stormforge-system`. A Stormforge publica o complemento.

Para obter informações sobre o complemento, consulte Instalação do StormForge Agent na documentação da StormForge.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Splunk

O nome do complemento é `splunk_splunk-otel-collector-chart` e o namespace é `splunk-monitoring`. A Splunk publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento Splunk do Amazon EKS na documentação da Splunk.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Teleport

O nome do complemento é `teleport_teleport` e o namespace é `teleport`. A Teleport publica o complemento.

Para obter mais informações sobre o complemento, consulte Como o Teleport funciona na documentação do Teleport.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Tetrate

O nome do complemento é `tetrate-io_istio-distro` e o namespace é `istio-system`. A Tetrate Io publica o complemento.

Para obter informações sobre o complemento, consulte o site do Tetrate Istio Distro.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Upbound Universal Crossplane

O nome do complemento é `upbound_universal-crossplane` e o namespace é `upbound-system`. A Upbound publica o complemento.

Para obter informações sobre o complemento, consulte Upbound Universal Crossplane (UXP) na documentação da Upbound.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Upwind

O nome do complemento é `upwind` e o namespace é `upwind`. A Upwind publica o complemento.

Para obter mais informações sobre o complemento, consulte a Documentação da UpWind.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Criar um complemento do Amazon EKS

Os complementos do Amazon EKS são um software complementar para clusters do Amazon EKS. Todos os complementos do Amazon EKS:

- Incluem os patches de segurança e as correções de erros mais recentes.
- São validados pela AWS para funcionar com o Amazon EKS.
- Reduzem o trabalho necessário para gerenciar software complementar.

Você pode criar um complemento do Amazon EKS usando o `eksctl`, o AWS Management Console ou a AWS CLI. Se o complemento exigir um perfil do IAM, consulte os detalhes do complemento específico em Complementos do Amazon EKS para obter detalhes sobre como criar o perfil.

Pré-requisitos

Antes de criar um complemento, faça o seguinte:

- O cluster deverá existir para que você possa criar um complemento para ele. Para obter mais informações, consulte Criar um cluster do Amazon EKS..
- Verifique se o complemento exige um perfil do IAM. Para obter mais informações, consulte Verificar a compatibilidade da versão do complemento do Amazon EKS com um cluster.

- Verifique se a versão do complemento do Amazon EKS é compatível com o cluster. Para obter mais informações, consulte Verificar a compatibilidade da versão do complemento do Amazon EKS com um cluster.
- Verifique se a versão 0.187.0 ou superior da ferramenta da linha de comando do `eksctl` está instalada em seu computador ou no AWS CloudShell. Para obter mais informações, consulte Installation (Instalação) no site do `eksctl`.

Procedimento

Você pode criar um complemento do Amazon EKS usando o `eksctl`, o AWS Management Console ou a AWS CLI. Se o complemento exigir um perfil do IAM, consulte os detalhes do complemento específico em Complementos do Amazon EKS disponíveis da AWS para obter detalhes sobre como criar o perfil.

eksctl

Para criar um complemento do Amazon EKS usando o **`eksctl`**

1. Visualize os nomes dos complementos disponíveis para uma versão de cluster. Substitua *`1.30`* pela versão do cluster.

```
eksctl utils describe-addon-versions --kubernetes-version 1.30 | grep AddonName
```

Veja um exemplo de saída abaixo.

```
"AddonName": "aws-ebs-csi-driver",
                        "AddonName": "coredns",
                        "AddonName": "kube-proxy",
                        "AddonName": "vpc-cni",
                        "AddonName": "adot",
                        "AddonName": "dynatrace_dynatrace-operator",
                        "AddonName": "upbound_universal-crossplane",
                        "AddonName": "teleport_teleport",
                        "AddonName": "factorhouse_kpow",
                        [...]
```

2. Visualize as versões disponíveis para o complemento que você deseja criar. Substitua *`1.30`* pela versão do cluster. Substitua *`name-of-addon`* pelo nome do complemento

cujas versões você deseja visualizar. O nome deve ser um dos nomes retornados na etapa anterior.

```
eksctl utils describe-addon-versions --kubernetes-version 1.30 --name name-of-addon | grep AddonVersion
```

A saída a seguir é um exemplo do que é retornado para o complemento denominado `vpc-cni`. Você pode ver que o complemento tem várias versões disponíveis.

```
"AddonVersions": [
    "AddonVersion": "v1.12.0-eksbuild.1",
    "AddonVersion": "v1.11.4-eksbuild.1",
    "AddonVersion": "v1.10.4-eksbuild.1",
    "AddonVersion": "v1.9.3-eksbuild.1",
```

3. Determine se o complemento que você deseja criar é um complemento do Amazon EKS ou um complemento do AWS Marketplace. O AWS Marketplace tem complementos de terceiros que requerem que você conclua etapas adicionais para criar o complemento.

```
eksctl utils describe-addon-versions --kubernetes-version 1.30 --name name-of-addon | grep ProductUrl
```

Se nenhuma saída for retornada, o complemento será um complemento do Amazon EKS. Se for retornada uma saída, o complemento será um complemento do AWS Marketplace. A saída a seguir é para um complemento denominado `teleport_teleport`.

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?
sku=3bda70bb-566f-4976-806c-f96faef18b26"
```

Você pode saber mais sobre o complemento no AWS Marketplace com a URL retornada. Se o complemento exigir uma assinatura, você poderá fazer a assinatura do complemento por meio do AWS Marketplace. Se for criar um complemento no AWS Marketplace, a entidade principal do IAM que você está usando para criar o complemento deverá ter permissão para criar o perfil vinculado ao serviço AWSServiceRoleForAWSLicenseManagerRole. Para obter informações sobre como atribuir permissões a uma entidade do IAM, consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do IAM.

4. Crie um complemento do Amazon EKS. Copie o comando e substitua *user-data* da seguinte forma:

- Substitua o *my-cluster* pelo nome do cluster.
- Substitua *name-of-addon* pelo nome do complemento que você deseja criar.
- Se você quiser uma versão do complemento anterior à versão mais recente, substitua *latest* pelo número da versão que você deseja usar, retornado na saída de uma etapa anterior.
- Se o complemento usar um perfil da conta de serviço, substitua *111122223333* pela ID da conta e substitua *role-name* pelo nome do perfil. Para obter instruções para criar um perfil para a conta de serviço, consulte a documentação do complemento que você está criando. A especificação de um perfil da conta de serviço exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte Criar um provedor OIDC do IAM para o cluster.

  Se o complemento não usar um perfil da conta de serviço, exclua **--service-account-role-arn arn:aws:iam::111122223333:role/role-name**.
- Esse exemplo de comando sobrescreve a configuração de qualquer versão autogerenciada do complemento existente, se houver alguma. Se você não quiser sobrescrever a configuração de um complemento autogerenciado existente, remova a opção *--force*. Se você remover a opção e o complemento do Amazon EKS precisar sobrescrever a configuração de um complemento autogerenciado existente, a criação do complemento do Amazon EKS falhará com uma mensagem de erro para ajudar a resolver o conflito. Antes de especificar essa opção, certifique-se de que o complemento Amazon EKS não gerencie as configurações que você precisa gerenciar, pois essas configurações são substituídas por essa opção.

```
eksctl create addon --cluster my-cluster --name name-of-addon --version latest
 \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
force
```

Você pode ver uma lista de todas as opções disponíveis para o comando.

```
eksctl create addon --help
```

Para obter mais informações sobre as opções disponíveis, consulte Addons (Complementos) na documentação do `eksctl`.

AWS Management Console

Para criar um complemento do Amazon EKS usando a AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome do cluster para o qual deseja criar o complemento.
4. Escolha a guia Add-ons (Complementos).
5. Escolha Obter mais complementos.
6. Na página Select add-ons (Selecionar complementos), escolha os complementos que você deseja adicionar ao cluster. Você pode adicionar tantos complementos do Amazon EKS e complementos do AWS Marketplace quantos forem necessários.

   Para os complementos AWS Marketplace, o principal IAM que você está usando para criar o complemento deve ter permissões para ler as autorizações do complemento no AWS LicenseManager. AWS O LicenseManager requer o perfil vinculado ao serviço (SLR), AWSServiceRoleForAWSLicenseManagerRole que permite que os recursos da AWS gerenciem licenças em seu nome. O SLR é exigido uma única vez para cada conta, e você não precisará criar SLRs separados para cada complemento nem para cada cluster. Para obter informações sobre como atribuir permissões a uma entidade principal do IAM consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do IAM.

   Se os complementos do AWS Marketplace que você quer instalar não estiverem listados, será possível pesquisar os complementos do disponíveis inserindo texto na caixa de pesquisa. Nas opções de filtragem, você também pode filtrar por categoria, fornecedor ou modelo de preços e, em seguida, escolher os complementos nos resultados da pesquisa. Após selecionar os complementos que você deseja instalar, escolha Next (Avançar).
7. Na página Definir as configurações dos complementos selecionados:

   a. Escolha Exibir opções de assinatura para abrir o formulário Opções de assinatura. Revise as seções Detalhes de preços e Aspectos jurídicos e escolha o botão Inscrever-se para continuar.

   b. Em Versão, escolha a versão que você deseja usar. Recomendamos a versão marcada como latest (a mais recente), a menos que o complemento individual que você está

criando recomende outra versão. Para determinar se um complemento tem uma versão recomendada, consulte a documentação do complemento que você está criando.

c. Se todos os complementos que você selecionou mostrarem Exige assinatura em Status, selecione Próximo. Você não pode continuar a configurar esses complementos até fazer uma assinatura deles depois de criar o cluster. Para os complementos que não mostram Exige assinatura em Status, faça o seguinte:

   i. Em Select IAM role (Selecionar perfil do IAM), aceite a opção padrão, a menos que o complemento exija permissões do IAM. Se o complemento exigir permissões da AWS, você poderá usar o perfil do IAM do nó (Não definido) ou um perfil existente criado para ser usado com o complemento. Quando não há nenhum perfil para seleção, é porque você não tem nenhum perfil existente. Independentemente da opção escolhida, consulte a documentação do complemento que você está criando para criar uma política do IAM e anexá-la a um perfil. Para selecionar um perfil do IAM, você precisa ter um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte Criar um provedor OIDC do IAM para o cluster.

   ii. Escolha Optional configuration settings (Configurações opcionais).

   iii. Se o complemento exigir configuração, insira-a na caixa Configuration values (Valores de configuração). Para determinar se o complemento exige informações de configuração, consulte a documentação do complemento que você está criando.

   iv. Selecione uma das opções disponíveis em Método de resolução de conflitos. Se você escolher Substituir como Método de resolução de conflitos, uma ou mais configurações do complemento existente poderão ser substituídas pelas configurações do complemento do Amazon EKS. Se você não habilitar esta opção e houver um conflito com suas configurações existentes, a operação falhará. É possível usar a mensagem de erro resultante para solucionar o conflito. Antes de escolher essa opção, certifique-se de que o complemento do Amazon EKS não gerencie as configurações que você precisa autogerenciar.

   v. Escolha Próximo.

8. Na página Adicionar tags, escolha Criar. Depois que a instalação dos complementos for concluída, você verá os complementos instalados.

9. Se algum dos complementos que você instalou exigir uma assinatura, conclua as seguintes etapas:

a. Escolha o botão Subscribe (Assinar) no canto inferior direito do complemento. Você será levado para a página do complemento no AWS Marketplace. Leia as informações sobre o complemento, como a visão geral do produto e as informações sobre preços.

b. Selecione o botão Continue to Subscribe (Continuar a assinar) no canto superior direito da página do complemento.

c. Leia todos os Terms and Conditions (Termos e condições). Se você concordar com eles, escolha Accept Terms (Aceitar termos). O processamento da assinatura pode levar alguns minutos. Enquanto a assinatura estiver sendo processada, o botão Return to Amazon EKS Console (Retornar ao console do Amazon EKS) estará acinzentado.

d. Depois que o processamento da assinatura for concluído, o botão Return to Amazon EKS Console (Retornar ao console do Amazon EKS) não estará mais acinzentado. Escolha o botão para voltar à guia Add-ons (Complementos) do console Amazon EKS para o cluster.

e. Para o complemento que você assinou, escolha Remove and reinstall (Remover e reinstalar) e depois escolha Reinstall add-on(Reinstalar complemento). A instalação do complemento pode levar vários minutos. Quando a instalação estiver concluída, você poderá configurar o complemento.

AWS CLI

Pré-requisito

A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.

Para criar um complemento do Amazon EKS usando a AWS CLI

1. Determine quais complementos estão disponíveis. Você pode ver todos os complementos disponíveis, seu tipo e seu editor. Você também pode ver o URL dos complementos disponíveis por meio do AWS Marketplace. Substitua *1.30* pela versão do cluster.

```
aws eks describe-addon-versions --kubernetes-version 1.30 \
    --query 'addons[].{MarketplaceProductUrl: marketplaceInformation.productUrl,
 Name: addonName, Owner: owner Publisher: publisher, Type: type}' --output table
```

Veja um exemplo de saída abaixo.

```
-------------------------------------------------------------------------------------------
|
 DescribeAddonVersions
       |
+-----------------------------------------------------------
+------------------------------+------------------+--------------
+---------------------+
|                      MarketplaceProductUrl                     |
 Name               |      Owner       |  Publisher   |        Type         |
+-----------------------------------------------------------
+------------------------------+------------------+--------------
+---------------------+
|  None                                                          |  aws-ebs-csi-
driver           |  aws             |  eks         |  storage            |
|  None                                                          |  coredns
                |  aws             |  eks         |  networking         |
|  None                                                          |  kube-proxy
                |  aws             |  eks         |  networking         |
|  None                                                          |  vpc-cni
                |  aws             |  eks         |  networking         |
|  None                                                          |  adot
                |  aws             |  eks         |  observability      |
|  https://aws.amazon.com/marketplace/pp/prodview-brb73nceicv7u |
 dynatrace_dynatrace-operator |  aws-marketplace |  dynatrace   |  monitoring
      |
|  https://aws.amazon.com/marketplace/pp/prodview-uhc2iwi5xysoc |
 upbound_universal-crossplane |  aws-marketplace |  upbound     |  infra-
management   |
```

```
|  https://aws.amazon.com/marketplace/pp/prodview-hd2ydsrgqy4li |
 teleport_teleport            |  aws-marketplace |  teleport    |  policy-
management  |
|  https://aws.amazon.com/marketplace/pp/prodview-vgghgqdsplhvc |
 factorhouse_kpow             |  aws-marketplace |  factorhouse |  monitoring
      |
|  [...]                                                       |  [...]
                |  [...]           |  [...]       |  [...]              |
+-------------------------------------------------------------
+------------------------------+------------------+--------------
+--------------------+
```

A sua saída pode ser diferente. Neste exemplo de saída, há três complementos diferentes disponíveis do tipo `networking` e cinco complementos com um editor do tipo `eks`. Os complementos com `aws-marketplace` na coluna `Owner` podem requerer uma assinatura antes que você possa instalá-los. Você pode visitar a URL para obter mais informações e fazer uma assinatura do complemento.

2. Você pode ver quais versões estão disponíveis para cada complemento. Substitua *`1.30`* pela versão do seu cluster e substitua *`vpc-cni`* pelo nome de um complemento retornado na etapa anterior.

```
aws eks describe-addon-versions --kubernetes-version 1.30 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

Veja um exemplo de saída abaixo.

```
-----------------------------------------
|          DescribeAddonVersions        |
+-----------------+---------------------+
| Defaultversion  |       Version       |
+-----------------+---------------------+
|  False          |  v1.12.0-eksbuild.1 |
|  True           |  v1.11.4-eksbuild.1 |
|  False          |  v1.10.4-eksbuild.1 |
|  False          |  v1.9.3-eksbuild.1  |
+-----------------+---------------------+
```

A versão com `True` na coluna `Defaultversion` é a versão com a qual o complemento foi criado, por padrão.

3. (Opcional) Encontre as opções de configuração para o complemento escolhido executando o seguinte comando:

```
aws eks describe-addon-configuration --addon-name vpc-cni --addon-
version v1.12.0-eksbuild.1
```


```
{
    "addonName": "vpc-cni",
    "addonVersion": "v1.12.0-eksbuild.1",
    "configurationSchema": "{\"$ref\":\"#/definitions/VpcCni\",\"$schema
\":\"http://json-schema.org/draft-06/schema#\",\"definitions\":{\"Cri\":
{\"additionalProperties\":false,\"properties\":{\"hostPath\":{\"$ref\":
\"#/definitions/HostPath\"}},\"title\":\"Cri\",\"type\":\"object\"},\"Env
\":{\"additionalProperties\":false,\"properties\":{\"ADDITIONAL_ENI_TAGS
\":{\"type\":\"string\"},\"AWS_VPC_CNI_NODE_PORT_SUPPORT\":{\"format\":
\"boolean\",\"type\":\"string\"},\"AWS_VPC_ENI_MTU\":{\"format\":\"integer
\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CONFIGURE_RPFILTER\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG\":
{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_EXTERNALSNAT
\":{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOGLEVEL
\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOG_FILE\":{\"type
\":\"string\"},\"AWS_VPC_K8S_CNI_RANDOMIZESNAT\":{\"type\":
\"string\"},\"AWS_VPC_K8S_CNI_VETHPREFIX\":{\"type\":\"string
\"},\"AWS_VPC_K8S_PLUGIN_LOG_FILE\":{\"type\":\"string\"},
\"AWS_VPC_K8S_PLUGIN_LOG_LEVEL\":{\"type\":\"string\"},\"DISABLE_INTROSPECTION
\":{\"format\":\"boolean\",\"type\":\"string\"},\"DISABLE_METRICS\":{\"format
\":\"boolean\",\"type\":\"string\"},\"DISABLE_NETWORK_RESOURCE_PROVISIONING
\":{\"format\":\"boolean\",\"type\":\"string\"},\"ENABLE_POD_ENI\":{\"format
\":\"boolean\",\"type\":\"string\"},\"ENABLE_PREFIX_DELEGATION\":{\"format
\":\"boolean\",\"type\":\"string\"},\"WARM_ENI_TARGET\":{\"format\":\"integer
\",\"type\":\"string\"},\"WARM_PREFIX_TARGET\":{\"format\":\"integer\",
\"type\":\"string\"}},\"title\":\"Env\",\"type\":\"object\"},\"HostPath\":
{\"additionalProperties\":false,\"properties\":{\"path\":{\"type\":\"string\"}},
\"title\":\"HostPath\",\"type\":\"object\"},\"Limits\":{\"additionalProperties
\":false,\"properties\":{\"cpu\":{\"type\":\"string\"},\"memory\":{\"type
\":\"string\"}},\"title\":\"Limits\",\"type\":\"object\"},\"Resources\":
{\"additionalProperties\":false,\"properties\":{\"limits\":{\"$ref\":\"#/
definitions/Limits\"},\"requests\":{\"$ref\":\"#/definitions/Limits\"}},
\"title\":\"Resources\",\"type\":\"object\"},\"VpcCni\":{\"additionalProperties
\":false,\"properties\":{\"cri\":{\"$ref\":\"#/definitions/Cri\"},\"env\":
{\"$ref\":\"#/definitions/Env\"},\"resources\":{\"$ref\":\"#/definitions/
Resources\"}},\"title\":\"VpcCni\",\"type\":\"object\"}}}"
```

```
}
```

A saída é um esquema JSON padrão.

Aqui está um exemplo de valores de configuração válidos, no formato JSON, que funcionam com o esquema acima.

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
}
```

Aqui está um exemplo de valores de configuração válidos, no formato YAML, que funcionam com o esquema acima.

```
  resources:
    limits:
      cpu: 100m
```

4. Determine se o complemento requer permissões do IAM. Nesse caso, você precisa (1) determinar se deseja usar identidades de pods do EKS ou perfis do IAM para contas de serviço (IRSA), (2) determinar o ARN do perfil do IAM a ser usado com o complemento e (3) determinar o nome da conta do serviço Kubernetes usada pelo complemento. É possível encontrar essas informações na documentação ou usando a API da AWS. Consulte Recuperar informações do IAM sobre um complemento.

   - O Amazon EKS sugerirá o uso de Identidades de Pods do EKS se o complemento oferecer suporte a esse recurso. Isso requer que o Agente de Identidade de Pods esteja instalado no seu cluster. Para obter mais informações sobre como usar Identidades de Pods com complementos, consulte Perfis do IAM para complementos do Amazon EKS.
   - Se o complemento ou seu cluster não estiver configurado para Identidades de Pods do EKS, use o IRSA. Confirme se o IRSA está configurado no seu cluster.
   - Analise a documentação dos complementos do Amazon EKS para determinar se o complemento exige permissões do IAM e o nome da conta do serviço Kubernetes associada.

5. Crie um complemento do Amazon EKS. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado:

   - Substitua o *my-cluster* pelo nome do cluster.
   - Substitua *vpc-cni* pelo nome de complemento que você deseja criar, retornado na saída da etapa anterior.
   - Substitua *version-number* pela versão que você deseja usar, retornada na saída da etapa anterior.
   - Se não houver necessidade de permissões do IAM, exclua *<service-account-configuration>*.
   - Execute um destes procedimentos:
     - Se o complemento (1) exigir permissões do IAM e (2) seu cluster usar Identidades de Pods do EKS, substitua *<service-account-configuration>* pela associação de identidade de pods a seguir. Substitua *<service-account-name>* pelo nome da conta de serviço usada pelo complemento. Substitua *<role-arn>* pelo ARN de um perfil do IAM. O perfil deve ter a política de confiança exigida pelas Identidades de Pods do EKS.

       ```
       --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
       ```

     - Se o complemento (1) exigir permissões do IAM e (2) seu cluster usar o IRSA, substitua *<service-account-configuration>* pela seguinte configuração do IRSA. Substitua *111122223333* pelo ID da sua conta e *role-name* pelo nome do perfil do IAM existente que você criou. Para obter instruções para criar o perfil, consulte a documentação do complemento que você está criando. A especificação de um perfil da conta de serviço exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte Criar um provedor OIDC do IAM para o cluster.

       ```
       --service-account-role-arn arn:aws::iam::111122223333:role/role-name
       ```

   - Esses comandos de exemplo substituem a opção `--configuration-values` de qualquer versão autogerenciada existente do complemento, se houver. Substitua isso pelos valores de configuração desejados, como uma string ou uma entrada de arquivo. Se você não deseja fornecer valores de configuração, exclua a opção `--configuration-`

`values`. Se você não quiser que a AWS CLI sobrescreva a configuração de um complemento autogerenciado existente, remova a opção *`--resolve-conflicts OVERWRITE`*. Se você remover a opção e o complemento do Amazon EKS precisar sobrescrever a configuração de um complemento autogerenciado existente, a criação do complemento do Amazon EKS falhará com uma mensagem de erro para ajudar a resolver o conflito. Antes de especificar essa opção, certifique-se de que o complemento Amazon EKS não gerencie as configurações que você precisa gerenciar, pois essas configurações são substituídas por essa opção.

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
    <service-account-configuration> --configuration-values '{"resources":
{"limits":{"cpu":"100m"}}}' --resolve-conflicts OVERWRITE
```

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
    <service-account-configuration> --configuration-values 'file://example.yaml'
 --resolve-conflicts OVERWRITE
```

Para obter uma lista completa das opções disponíveis, consulte `create-addon` na Amazon EKS Command Line Reference (Referência da linha de comando do Amazon EKS). Se o complemento que você criou tiver `aws-marketplace` listado na coluna `Owner` de uma etapa anterior, a criação poderá falhar e você poderá receber uma mensagem de erro semelhante a que se segue.

```
{
    "addon": {
        "addonName": "addon-name",
        "clusterName": "my-cluster",
        "status": "CREATE_FAILED",
        "addonVersion": "version",
        "health": {
            "issues": [
                {
                    "code": "AddonSubscriptionNeeded",
                    "message": "You are currently not subscribed to this add-
on. To subscribe, visit the AWS Marketplace console, agree to the seller EULA,
 select the pricing type if required, then re-install the add-on"
```

```
[...]
```

Se você receber um erro semelhante ao erro da saída anterior, visite a URL na saída de uma etapa anterior para assinar o complemento. Depois de fazer a assinatura, execute o comando `create-addon` novamente.

Atualizar um complemento do Amazon EKS

O Amazon EKS não atualiza automaticamente o complemento quando novas versões são lançadas nem depois que você atualiza o cluster para uma nova versão secundária do Kubernetes. Para atualizar um complemento para um cluster existente, você deve iniciar a atualização. Após a atualização ser iniciada, o Amazon EKS atualizará o complemento do Amazon EKS para você. Antes de atualizar um complemento, revise sua documentação atual. Para obter uma lista dos complementos disponíveis, consulte Complementos do Amazon EKS disponíveis da AWS. Se o complemento exigir um perfil do IAM, consulte os detalhes do complemento específico em Complementos do Amazon EKS disponíveis da AWS para obter detalhes sobre como criar o perfil.

Pré-requisitos

Antes de criar um complemento, faça o seguinte:

- Verifique se o complemento exige um perfil do IAM. Para obter mais informações, consulte Complementos do Amazon EKS.
- Verifique se a versão do complemento do Amazon EKS é compatível com o cluster. Para obter mais informações, consulte Verificar a compatibilidade da versão do complemento do Amazon EKS com um cluster.

Procedimento

Você pode atualizar um complemento do Amazon EKS usando `eksctl`, o AWS Management Console ou a AWS CLI.

eksctl

Para atualizar um complemento do Amazon EKS usando o **`eksctl`**

1. Determine os complementos e as versões dos complementos instalados atualmente no cluster. Substitua o *`my-cluster`* pelo nome do cluster.

```
eksctl get addon --cluster my-cluster
```

Veja um exemplo de saída abaixo.

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0                v1.23.8-eksbuild.2
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0                v1.12.0-
eksbuild.1,v1.11.4-eksbuild.1,v1.11.3-eksbuild.1,v1.11.2-eksbuild.1,v1.11.0-
eksbuild.1
```

Sua saída pode ser diferente, dependendo dos complementos e versões que você tiver no cluster. Você pode ver que, na saída do exemplo anterior, dois complementos existentes no cluster têm versões mais recentes disponíveis na coluna `UPDATE AVAILABLE`.

2. Atualize o complemento.

   a. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário:

      - Substitua o *my-cluster* pelo nome do cluster.
      - Substitua *region-code* pela Região da AWS em que seu cluster está localizado.
      - Substitua *vpc-cni* pelo nome de complemento que você deseja atualizar, retornado na saída da etapa anterior.
      - Se você quiser uma versão do complemento anterior à versão mais recente, substitua *latest* pelo número de versão que você deseja usar, retornado na saída da etapa anterior. Alguns complementos têm versões recomendadas. Para obter mais informações, consulte a documentação do complemento que você está atualizando.
      - Se o complemento usar uma conta de serviço Kubernetes e um perfil do IAM, substitua *111122223333* pelo ID da sua conta e *role-name* pelo nome de um perfil do função IAM existente que você criou. Para obter instruções para criar o perfil, consulte a documentação do complemento que você está criando. A especificação de um perfil da conta de serviço exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte Criar um provedor OIDC do IAM para o cluster.

Se o complemento não usar uma conta de serviço do Kubernetes e um perfil do IAM, exclua a linha **`serviceAccountRoleARN: arn:aws:iam::111122223333:role/role-name`**.

- A opção *`preserve`* mantém os valores existentes para o complemento. Se você definiu valores personalizados para as configurações do complemento e não usar essa opção, o Amazon EKS sobrescreverá seus valores pelos valores padrão. Se você usar essa opção, recomendamos testar qualquer alteração de campo e valor em um cluster que não seja de produção antes de atualizar o complemento no cluster de produção. Se você alterar esse valor para `overwrite`, todas as configurações serão alteradas para os valores padrão do Amazon EKS. Se você definiu valores personalizados para qualquer configuração, eles poderão ser sobrescritos pelos valores padrão do Amazon EKS. Se você alterar esse valor para `none`, o Amazon EKS não alterará o valor de nenhuma configuração, mas a atualização poderá falhar. Se a atualização falhar, você receberá uma mensagem de erro para ajudar a resolver o conflito.

```
cat >update-addon.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code

addons:
- name: vpc-cni
  version: latest
  serviceAccountRoleARN: arn:aws:iam::111122223333:role/role-name
  resolveConflicts: preserve
EOF
```

b. Execute o comando modificado para criar o arquivo `update-addon.yaml`.

c. Aplique o arquivo de configuração ao cluster.

```
eksctl update addon -f update-addon.yaml
```

Para obter mais informações sobre atualização de complementos, consulte Atualizar complementos na documentação do `eksctl`.

AWS Management Console

Para atualizar o complemento do Amazon EKS usando o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome do cluster para o qual você deseja atualizar o complemento..
4. Escolha a guia Add-ons (Complementos).
5. Escolha o complemento que deseja atualizar.
6. Selecione a opção Editar.
7. Na página Configure ***`nome do complemento`***, faça o seguinte:
   a. Selecione a Versão que você deseja usar. O complemento pode ter uma versão recomendada. Para obter mais informações, consulte a documentação do complemento que você está atualizando.
   b. Em Selecionar perfil do IAM, é possível usar o perfil do IAM do nó (Não definido) ou um perfil existente criado para ser usado com o complemento. Quando não há nenhum perfil para seleção, é porque você não tem nenhum perfil existente. Independentemente da opção escolhida, consulte a documentação do complemento que você está criando para criar uma política do IAM e anexá-la a um perfil. Para selecionar um perfil do IAM, você precisa ter um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte Criar um provedor OIDC do IAM para o cluster.
   c. Expanda Definições de configuração opcionais.
   d. Em Valores de configuração, insira informações de configuração específica do complemento. Para obter mais informações, consulte a documentação do complemento que você está atualizando.
   e. Em Conflict resolution method (Método de resolução de conflitos), selecione uma das opções. Se você definiu valores personalizados para as configurações do complemento, recomendamos escolher a opção Preserve (Manter). Se você não escolher essa opção, o Amazon EKS sobrescreverá seus valores com os valores padrão. Se você usar essa opção, recomendamos testar qualquer alteração de campo e valor em um cluster que não seja de produção antes de atualizar o complemento no cluster de produção.
8. Escolha Salvar alterações.

AWS CLI

Pré-requisito

A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.

Para atualizar o complemento do Amazon EKS usando o AWS CLI

1. Veja uma lista de complementos instalados. Substitua o *`my-cluster`* pelo nome do cluster.

```
aws eks list-addons --cluster-name my-cluster
```

   Veja um exemplo de saída abaixo.

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni"
    ]
}
```

2. Visualize a versão atual do complemento que você deseja atualizar. Substitua *`my-cluster`* pelo nome do cluster e *`vpc-cni`* pelo nome do complemento que você deseja atualizar.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

   Veja um exemplo de saída abaixo.

```
v1.10.4-eksbuild.1
```

3. Determine quais versões do estão disponíveis para a versão do cluster. Substitua *1.30* pela versão do cluster e *vpc-cni* pelo nome do complemento que você deseja atualizar.

```
aws eks describe-addon-versions --kubernetes-version 1.30 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

Veja um exemplo de saída abaixo.

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+---------------------+
| Defaultversion  |       Version       |
+-----------------+---------------------+
|  False          |  v1.12.0-eksbuild.1 |
|  True           |  v1.11.4-eksbuild.1 |
|  False          |  v1.10.4-eksbuild.1 |
|  False          |  v1.9.3-eksbuild.1  |
+-----------------+---------------------+
```

A versão com `True` na coluna `Defaultversion` é a versão com a qual o complemento foi criado, por padrão.

4. Atualize seu complemento. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado. Para obter mais informações sobre este comando, consulte `update-addon` na Referência da linha de comando do Amazon EKS.

- Substitua o *my-cluster* pelo nome do cluster.
- Substitua *vpc-cni* pelo nome de complemento que você deseja atualizar, que foi retornado na saída da etapa anterior.
- Substitua *version-number* pela versão para a qual você deseja atualizar, retornada na saída da etapa anterior. Alguns complementos têm versões recomendadas. Para obter mais informações, consulte a documentação do complemento que você está atualizando.
- Se o complemento usar uma conta de serviço Kubernetes e um perfil do IAM, substitua *111122223333* pelo ID da sua conta e *role-name* pelo nome de um perfil do função IAM existente que você criou. Para obter instruções para criar o perfil, consulte a documentação do complemento que você está criando. A especificação de um perfil da conta de serviço exige que você tenha um provedor OpenID Connect (OIDC) do IAM para

o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte Criar um provedor OIDC do IAM para o cluster.

Se o complemento não usar uma conta de serviço do Kubernetes e um perfil do IAM, exclua a linha **`serviceAccountRoleARN: arn:aws:iam::111122223333:role/role-name`**.

- A opção *PRESERVE* (MANTER) de **`--resolve-conflicts`** mantém os valores existentes para o complemento. Se você definiu valores personalizados para as configurações do complemento e não usar essa opção, o Amazon EKS sobrescreverá seus valores pelos valores padrão. Se você usar essa opção, recomendamos testar qualquer alteração de campo e valor em um cluster que não seja de produção antes de atualizar o complemento no cluster de produção. Se você alterar esse valor para `OVERWRITE`, todas as configurações serão alteradas para os valores padrão do Amazon EKS. Se você definiu valores personalizados para qualquer configuração, eles poderão ser sobrescritos pelos valores padrão do Amazon EKS. Se você alterar esse valor para `NONE`, o Amazon EKS não alterará o valor de nenhuma configuração, mas a atualização poderá falhar. Se a atualização falhar, você receberá uma mensagem de erro para ajudar a resolver o conflito.

- Se você deseja remover todas as configurações personalizadas, execute a atualização usando a opção *`--configuration-values '{}'`*. Isso definirá todas as configurações personalizadas aos valores padrão. Se você não deseja alterar a configuração personalizada, não forneça o sinalizador *`--configuration-values`*. Se você deseja ajustar uma configuração personalizada, substitua *`{}`* pelos novos parâmetros. Para ver uma lista de parâmetros, consulte a etapa visualizar esquema de configuração na seção sobre a criação de um complemento.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
configuration-values '{}' --resolve-conflicts PRESERVE
```

5. Verifique o status da atualização. Substitua *`my-cluster`* pelo nome do cluster e *`vpc-cni`* pelo nome do complemento que você está atualizando.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

Veja um exemplo de saída abaixo.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "UPDATING",
[...]
```

Quando a atualização estiver concluída, o status será `ACTIVE`.

Verificar a compatibilidade da versão do complemento do Amazon EKS com um cluster

Antes de criar um complemento do Amazon EKS, é necessário verificar se a versão do complemento do Amazon EKS é compatível com seu cluster.

Use a API `describe-addon-verisions` para listar as versões disponíveis dos complementos do EKS e quais versões do Kubernetes são compatíveis com cada versão do complemento.

Verificar a compatibilidade do complemento (AWS CLI)

1. Verifique se a AWS CLI está instalada e funcionando com o `aws sts get-caller-identity`. Se esse comando não funcionar, saiba como Começar com o AWS CLI.
2. Determine o nome do complemento para o qual você deseja recuperar as informações de compatibilidade de versão, por exemplo, `amazon-cloudwatch-observability`.
3. Determine a versão do Kubernetes do seu cluster, por exemplo, `1.28`.
4. Use a AWS CLI para recuperar as versões do complemento que são compatíveis com a versão do Kubernetes do seu cluster.

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --
kubernetes-version 1.29
```

Veja um exemplo de saída abaixo.

```
{
    "addons": [
```

```
        {
            "addonName": "amazon-cloudwatch-observability",
            "type": "observability",
            "addonVersions": [
                {
                    "addonVersion": "v1.5.0-eksbuild.1",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.28",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
[...]
```

Essa saída mostra que a versão do complemento `v1.5.0-eksbuild.1` é compatível com o cluster do Kubernetes versão `1.28`.

Remover um complemento do Amazon EKS de um cluster

É possível remover o complemento do Amazon EKS do cluster via `eksctl`, AWS Management Console ou AWS CLI.

Quando um complemento do Amazon EKS é removido de um cluster:

- Não há tempo de inatividade para a funcionalidade que o complemento fornece.
- Se estiver usando Perfis do IAM para contas de serviço (IRSA) e não houver um perfil do IAM associado ao complemento, o perfil do IAM não será removido.
- Se você estiver usando identidades de pods, todas as associações de identidades de pods pertencentes ao complemento serão removidas. Se você especificar a opção `--preserve` para a AWS CLI, as associações serão preservadas.
- O Amazon EKS deixará de gerenciar as configurações do complemento.
- O console deixará de notificar você quando novas versões estão disponíveis.

- Você não poderá atualizar o complemento usando nenhuma das ferramentas ou APIs da AWS.
- Você pode optar por deixar o software complementar no cluster para que ele seja autogerenciado ou pode remover o software complementar do cluster. Você só deverá remover o software complementar do cluster se não houver nenhum recurso no cluster que dependa da funcionalidade fornecida pelo complemento.

Pré-requisitos

Antes de criar um complemento, faça o seguinte:

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS.
- Verifique se o complemento exige um perfil do IAM. Para obter mais informações, consulte
- Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Procedimento

É possível remover um complemento do Amazon EKS via `eksctl`, AWS Management Console ou AWS CLI. Se o complemento exigir um perfil do IAM, consulte os detalhes do complemento específico em Complementos do Amazon EKS disponíveis da AWS para obter detalhes sobre como criar o perfil.

Você tem duas opções ao remover um complemento do Amazon EKS.

- Preserve add-on software on your cluster (Preservar software de complemento no cluster): esta opção remove o gerenciamento do Amazon EKS de qualquer configuração. Também remove a capacidade do Amazon EKS de notificar você sobre atualizações e de atualizar automaticamente o complemento do Amazon EKS depois de iniciar uma atualização. No entanto, ele preserva o software de complemento em seu cluster. Essa opção torna o complemento em uma instalação autogerenciada, em vez de um complemento do Amazon EKS. Com essa opção, não há tempo de inatividade para o complemento.
- Remover completamente o software do complemento do cluster: recomendamos remover o complemento do Amazon EKS do cluster se não houver recursos no cluster que dependam dele.

eksctl

Para remover um complemento do Amazon EKS usando **`eksctl`**

1. Determine quais são os complementos instalados atualmente no cluster. Substitua o *`my-cluster`* pelo nome do cluster.

```
eksctl get addon --cluster my-cluster
```

Veja um exemplo de saída abaixo.

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0
[...]
```

Sua saída pode ser diferente, dependendo dos complementos e versões que você tiver no cluster.

2. Remova o complemento. Substitua *`my-cluster`* pelo nome do cluster e *`name-of-add-on`* pelo nome do complemento que você deseja remover, retornado na saída da etapa anterior. Se você remover a opção ***`--preserve`***, além de o Amazon EKS deixar de gerenciar o complemento, o software complementar será excluído do cluster.

```
eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
```

Para obter mais informações sobre a remoção de complementos, consulte Excluir complementos na documentação do `eksctl`.

AWS Management Console

Para remover o complemento do Amazon EKS via AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome do cluster para o qual você deseja remover o complemento do Amazon EKS.
4. Escolha a guia Add-ons (Complementos).

5. Escolha o complemento que deseja remover.
6. Escolha Remover.
7. Na caixa de diálogo de confirmação Remover: ***nome do complemento***, faça o seguinte:
    a. Se quiser que o Amazon EKS pare de gerenciar as configurações do complemento, selecione Preservar no cluster. Faça isso se quiser manter o software de complemento em seu cluster. Isso serve para que você possa gerenciar todas as configurações do complemento por conta própria.
    b. Insira o nome do complemento.
    c. Escolha Remover.

AWS CLI

Pré-requisito

Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Para remover o complemento do Amazon EKS via AWS CLI

1. Veja uma lista de complementos instalados. Substitua o *`my-cluster`* pelo nome do cluster.

```
aws eks list-addons --cluster-name my-cluster
```

Veja um exemplo de saída abaixo.

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni",
        "name-of-addon"
    ]
}
```

2. Remova o complemento instalado. Substitua *`my-cluster`* pelo nome do cluster e *`name-of-add-on`* pelo nome do complemento que você deseja remover. A remoção de ***`--preserve`*** remove o software do complemento do cluster.

```
aws eks delete-addon --cluster-name my-cluster --addon-name name-of-addon --
preserve
```

Veja o exemplo de saída abreviado abaixo.

```
{
    "addon": {
        "addonName": "name-of-add-on",
        "clusterName": "my-cluster",
        "status": "DELETING",
[...]
```

3. Verifique o status da remoção. Substitua *my-cluster* pelo nome do cluster e *name-of-addon* pelo nome do complemento que você está removendo.

```
aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
```

Após a exclusão do complemento, a saída de exemplo será semelhante a esta.

```
An error occurred (ResourceNotFoundException) when calling the DescribeAddon
 operation: No addon: name-of-addon found in cluster: my-cluster
```

Determinar os campos que podem ser personalizados para os complementos do Amazon EKS

Os complementos do Amazon EKS são instalados no cluster usando as configurações padrão de prática recomendada. Para obter mais informações sobre como adicionar um complemento do Amazon EKS ao cluster, consulte Complementos do Amazon EKS.

Talvez você queira personalizar a configuração de um complemento do Amazon EKS para habilitar recursos avançados. O Amazon EKS usa o recurso de aplicação no lado do servidor do Kubernetes para permitir o gerenciamento de um complemento pelo Amazon EKS sem substituir a configuração por configurações que não são gerenciadas pelo Amazon EKS. Para obter mais informações, consulte Server-Side Apply (Aplicação no lado do servidor) na documentação do Kubernetes. Para isso, o Amazon EKS gerencia um conjunto mínimo de campos para cada complemento instalado. Você pode modificar todos os campos que não forem gerenciados pelo Amazon EKS ou por outro

processo do ambiente de gerenciamento do Kubernetes, como `kube-controller-manager`, sem problemas.

> **Important**
>
> A modificação de um campo gerenciado pelo Amazon EKS impede que o Amazon EKS gerencie o complemento e pode resultar na substituição das alterações quando um complemento for atualizado.

Sintaxe do gerenciamento de campos

Quando você visualiza os detalhes de um objeto do Kubernetes, tanto os campos gerenciados quanto os não gerenciados são retornados na saída. Os campos gerenciados podem ser de um dos seguintes tipos:

- Fully managed (Totalmente gerenciado) – Todas as chaves para o campo são gerenciadas pelo Amazon EKS. Modificações em qualquer valor neste campo geram conflito.
- Partially managed (Parcialmente gerenciado) – Algumas chaves para o campo são gerenciadas pelo Amazon EKS. Somente modificações nas chaves gerenciadas explicitamente pelo Amazon EKS geram conflito.

Ambos os tipos de campos são marcados com `manager: eks`.

Cada chave é um `.` que representa o próprio campo, que sempre mapeia para um conjunto vazio, ou uma string que representa um subcampo ou item. A saída para o gerenciamento de campo consiste nos seguintes tipos de declarações:

- `f:`*`name`*, em que *`name`* é o nome de um campo em uma lista.
- `k:`*`keys`*, em que *`keys`* é um mapa dos campos do item em uma lista.
- `v:`*`value`*, em que *`value`* é o valor exato formatado em JSON do item de uma lista.
- `i:`*`index`*, em que *`index`* é a posição de um item na lista.

As seguintes partes do resultado para o complemento CoreDNS ilustram as declarações anteriores:

- Fully managed fields (Campos totalmente gerenciados) – Se um campo gerenciado tiver um (campo) `f:` especificado, mas não `k:` (chave), todo o campo é gerenciado. Modificações em quaisquer valores neste campo causam um conflito.

  No resultado a seguir, é possível ver que o contêiner chamado `coredns` é gerenciado pelo `eks`. Os subcampos `args`, `image` e `imagePullPolicy` também são gerenciados pelo `eks`. Modificações em qualquer valor nesse campo geram conflito.

  ```
  [...]
  f:containers:
    k:{"name":"coredns"}:
    .: {}
    f:args: {}
    f:image: {}
    f:imagePullPolicy: {}
  [...]
  manager: eks
  [...]
  ```

- Partially managed fields (campos parcialmente gerenciados) – Se uma chave gerenciada tiver um valor especificado, as chaves declaradas são gerenciadas para esse campo. A modificação das chaves especificadas gera conflito.

  No resultado a seguir, é possível ver que o `eks` gerencia os volumes `config-volume` e `tmp`, definidos com a chave `name`.

  ```
  [...]
  f:volumes:
    k:{"name":"config-volume"}:
      .: {}
      f:configMap:
        f:items: {}
        f:name: {}
      f:name: {}
    k:{"name":"tmp"}:
      .: {}
      f:name: {}
  [...]
  manager: eks
  [...]
  ```

- Adição de chaves em campos gerenciados parcialmente – Se apenas um valor de chave específico for gerenciado, você pode adicionar, com segurança, mais chaves, como, por exemplo, argumentos, em um campo sem gerar conflito. Se você adicionar outras chaves, certifique-se de que o campo não seja gerenciado primeiro. Adicionar ou modificar qualquer valor gerenciado causa um conflito.

  No resultado a seguir, é possível ver que tanto a chave do `name`, quando o `name` são gerenciados. Adicionar ou modificar qualquer nome de contêiner causa um conflito com essa chave gerenciada.

```
[...]
f:containers:
  k:{"name":"coredns"}:
[...]
    f:name: {}
[...]
manager: eks
[...]
```

Procedimento

Você pode usar o `kubectl` para ver quais campos são gerenciados pelo Amazon EKS para qualquer complemento do Amazon EKS.

Você pode modificar todos os campos que não forem gerenciados pelo Amazon EKS ou por outro processo do ambiente de gerenciamento do Kubernetes, como `kube-controller-manager`, sem problemas.

1. Determine qual complemento você deseja examinar. Para ver todas as `deployments` e os `DaemonSets` implantados no cluster, consulte Visualizar os recursos do Kubernetes.
2. Exiba os campos gerenciados para um complemento executando o seguinte comando:

```
kubectl get type/add-on-name -n add-on-namespace -o yaml
```

   Por exemplo, você pode ver os campos gerenciados do complemento CoreDNS com o comando a seguir.

```
kubectl get deployment/coredns -n kube-system -o yaml
```

O gerenciamento de campo é listado na seção a seguir no resultado retornado.

```
[...]
managedFields:
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
[...]
```

> **Note**
>
> Se a saída não exibir `managedFields`, adicione **`--show-managed-fields`** ao comando e execute-o novamente. A versão do `kubectl` que você está usando determina se os campos gerenciados serão retornados por padrão.

Próximas etapas

Personalize os campos não pertencentes à AWS para seu complemento.

Perfis do IAM para complementos do Amazon EKS

Alguns complementos do Amazon EKS precisam de permissões e perfis do IAM para chamar APIs da AWS. Por exemplo, o complemento CNI da Amazon VPC chama determinadas APIs da AWS para configurar recursos de rede na sua conta. Esses complementos precisam receber permissão usando o IAM. Mais especificamente, a conta de serviço do pod que executa o complemento precisa estar associada a um perfil do IAM com uma política do IAM específica.

A forma recomendada de conceder permissões da AWS para workloads de cluster é usando o recurso Identidades de Pods do Amazon EKS. Você pode usar uma Associação de identidade de pods para mapear a conta de serviço de um complemento para um perfil do IAM. Se um pod usa uma conta de serviço que tem uma associação, o Amazon EKS define variáveis de ambiente nos contêineres do pod. As variáveis de ambiente configuram os SDKs da AWS, incluindo a AWS CLI, para usar as credenciais de Identidade de pods do EKS. Para obter mais informações, consulte Saiba mais sobre as identidades de pods do EKS.

Os complementos do Amazon EKS podem ajudar a gerenciar o ciclo de vida das associações de identidades de pods correspondentes ao complemento. Por exemplo, você pode criar ou atualizar

um complemento do Amazon EKS e a associação de identidade de pods necessária em uma única chamada de API. O Amazon EKS também fornece uma API para recuperar políticas do IAM sugeridas.

Uso sugerido:

1. Confirme se o agente de identidade de pods do Amazon EKS está configurado no cluster.
2. Determine se o complemento que você deseja instalar requer permissões do IAM usando a operação `describe-addon-versions` da AWS CLI. Se o sinalizador `requiresIamPermissions` for `true`, você deverá usar a operação `describe-addon-configurations` para determinar as permissões necessárias para o complemento. A resposta inclui uma lista sugerida de políticas do IAM gerenciadas.
3. Recupere o nome da conta do serviço Kubernetes e a política do IAM sugerida usando a operação `describe-addon-configuration` da CLI. Avalie o escopo da política sugerida em relação aos seus requisitos de segurança.
4. Crie um perfil do IAM usando a política de permissões sugerida e a política de confiança exigida pela Identidade de Pod. Para obter mais informações, consulte Criar uma associação ao EKS Pod Identity.
5. Crie ou atualize um complemento do Amazon EKS usando a CLI. Especifique pelo menos uma associação de identidade de pods. Uma associação de identidade de pods é o nome de uma conta de serviço do Kubernetes e o ARN do perfil do IAM.

Considerações:

- Associações de identidades de pods criadas usando as APIs do complemento pertencem ao respectivo complemento. Se você excluir o complemento, a associação de identidade de pods também será excluída. É possível evitar essa exclusão em cascata usando a opção `preserve` ao excluir um complemento usando a AWS CLI ou a API. Também é possível atualizar ou excluir diretamente a associação de identidade de pods, se necessário. Os complementos não podem assumir a propriedade de associações de identidades de pods existentes. Você deve excluir a associação existente e recriá-la usando uma operação de criação ou atualização de complemento.
- O Amazon EKS recomenda o uso de associações de identidades de pods para gerenciar permissões do IAM para complementos. O método anterior, perfis do IAM para contas de serviço (IRSA), ainda é compatível. Você pode especificar um `serviceAccountRoleArn` do IRSA e uma associação de identidade de pods para um complemento. Se o agente de identidade de pods do EKS estiver instalado no cluster, `serviceAccountRoleArn` será ignorado, e o EKS usará

a associação de identidade de pods fornecida. Se a Identidade de Pods não estiver habilitada, `serviceAccountRoleArn` será usado.

- Se você atualizar as associações de identidades de pods para um complemento existente, o Amazon EKS iniciará uma reinicialização contínua dos pods do complemento.

Recuperar informações do IAM sobre um complemento do Amazon EKS

Antes de criar um complemento, use a AWS CLI para determinar:

- Se o complemento requer permissões do IAM
- A política de IAM sugerida para uso

Recuperar informações do IAM sobre um complemento do Amazon EKS (AWS CLI)

1. Determinar o nome do complemento que você deseja instalar e a versão do Kubernetes do seu cluster. Para obter mais informações sobre complementos, consulte Complementos do Amazon EKS.
2. Use a AWS CLI para determinar se o add-on requer permissões de IAM.

   ```
   aws eks describe-addon-versions \
   --addon-name <addon-name> \
   --kubernetes-version <kubernetes-version>
   ```

   Por exemplo:

   ```
   aws eks describe-addon-versions \
   --addon-name aws-ebs-csi-driver \
   --kubernetes-version 1.30
   ```

   Analise o seguinte exemplo de saída. Observe que `requiresIamPermissions` é `true` e a versão padrão do complemento. Você precisa especificar a versão do complemento ao recuperar a política do IAM recomendada.

   ```
   {
       "addons": [
           {
               "addonName": "aws-ebs-csi-driver",
   ```

```
            "type": "storage",
            "addonVersions": [
                {
                    "addonVersion": "v1.31.0-eksbuild.1",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.30",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                    "requiresConfiguration": false,
                    "requiresIamPermissions": true
                },
[...]
```

3. Se o complemento exigir permissões do IAM, use a AWS CLI para recuperar uma política do IAM recomendada.

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name <addon-name> \
--addon-version <addon-version>
```

Por exemplo:

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name aws-ebs-csi-driver \
--addon-version v1.31.0-eksbuild.1
```

Analise a seguinte saída. Anote o `recommendedManagedPolicies`.

```
[
    {
        "serviceAccount": "ebs-csi-controller-sa",
```

```
        "recommendedManagedPolicies": [
            "arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"
        ]
    }
]
```

4. Crie um perfil do IAM e anexe a Política gerenciada recomendada. Como alternativa, revise a política gerenciada e defina o escopo das permissões conforme apropriado. Para mais informações, consulte Criar uma associação ao EKS Pod Identity.

Usar identidades de Pods para atribuir um perfil do IAM a um complemento do Amazon EKS

Alguns complementos do Amazon EKS precisam de permissões e perfis do IAM. Antes de adicionar e atualizar um complemento do Amazon EKS para usar uma associação de identidade de pods, verifique o perfil e a política a serem usados. Para obter mais informações, consulte Recuperar informações do IAM sobre um complemento do Amazon EKS.

Atualizar um complemento do Amazon EKS para usar uma associação de identidade de pods (AWS CLI)

1. Determine:
   - `cluster-name`: o nome do cluster no qual o complemento será instalado.
   - `addon-name`: o nome do complemento a ser instalado.
   - `service-account-name`: o nome da conta do serviço Kubernetes usada pelo complemento.
   - `iam-role-arn`: o ARN de um perfil do IAM com permissões suficientes para o complemento. O perfil deve ter a política de confiança necessária para a identidade de pods do EKS. Para ter mais informações, consulte Criar uma associação ao EKS Pod Identity.
2. Atualize o complemento usando a AWS CLI. Também é possível especificar associações de identidades de pods ao criar um complemento usando a mesma sintaxe `--pod-identity-assocations`. Observe que, quando você especifica associações de identidades de pods ao atualizar um complemento, todas as associações anteriores de identidades de pods são substituídas.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
```

```
--pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
```

Por exemplo:

```
aws eks update-addon --cluster-name mycluster \
--addon-name aws-ebs-csi-driver \
--pod-identity-associations 'serviceAccount=ebs-csi-controller-sa,roleArn=arn:aws:iam::123456789012:role/StorageDriver'
```

3. Valide se a associação de identidade de pods foi criada:

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

Se for bem-sucedido, você deve ver uma saída semelhante à seguinte. Observe o OwnerARN do complemento do EKS.

```
{
    "associations": [
        {
            "clusterName": "mycluster",
            "namespace": "kube-system",
            "serviceAccount": "ebs-csi-controller-sa",
            "associationArn": "arn:aws:eks:us-west-2:123456789012:podidentityassociation/mycluster/a-4wvljrezsukshq1bv",
            "associationId": "a-4wvljrezsukshq1bv",
            "ownerArn": "arn:aws:eks:us-west-2:123456789012:addon/mycluster/aws-ebs-csi-driver/9cc7ce8c-2e15-b0a7-f311-426691cd8546"
        }
    ]
}
```

Remover associações de identidades de pods de um complemento do Amazon EKS

Remova as associações de identidades de pods de um complemento do Amazon EKS.

Remover todas as associações de identidades de pods de um complemento do Amazon EKS (AWS CLI)

1. Determine:

- `cluster-name`: o nome do cluster EKS no qual instalar o complemento.
- `addon-name`: o nome do complemento do Amazon EKS a ser instalado.

2. Atualize o complemento para especificar uma matriz vazia de associações de identidades de pods.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations "[]"
```

Solucionar problemas de identidades de pods para complementos do EKS

Se os seus complementos estiverem encontrando erros ao tentar operações da API, do SDK ou da AWS CLI, verifique se:

- O Agente de identidade de pods está instalado no seu cluster.
  - Para obter informações sobre como instalar o Pod Identity Agent, consulte Configurar o Amazon EKS Pod Identity Agent.
- O complemento tem uma associação de identidade de pods válida.
  - Use o AWS CLI para recuperar as associações do nome da conta de serviço usada pelo complemento.

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

- O perfil do IAM tem a política de confiança necessária para as identidades de pods.
  - Use a AWS CLI para recuperar a política de confiança de um complemento.

```
aws iam get-role --role-name <role-name> --query Role.AssumeRolePolicyDocument
```

- O perfil do IAM tem as permissões necessárias para o complemento.
  - Use o AWS CloudTrail para avaliar eventos `AccessDenied` ou `UnauthorizedOperation`.
- O nome da conta de serviço na associação de identidade de pods corresponde ao nome da conta de serviço usado pelo complemento.
  - Para obter informações sobre os complementos disponíveis, consulte Complementos do Amazon EKS disponíveis da AWS.

Validar assinaturas de imagem de contêiner durante a implantação

Se você usa AWS Signer e deseja verificar imagens do contêiner assinadas no momento da implantação, você pode usar uma das seguintes soluções:

- Gatekeeper e Ratify — Use o Gatekeeper como controlador de admissão e o Ratify configurado com um AWS Signer plug-in como um web hook para validar assinaturas.
- Kyverno — Um mecanismo de política Kubernetes configurado com um AWS Signer plug-in para validar assinaturas.

> **Note**
>
> Antes de verificar as assinaturas das imagens do contêiner, configure o repositório confiável e a política de confiança do Notation, conforme exigido pelo controlador de admissão selecionado.

Executar treinamento de machine learning no Amazon EKS com Elastic Fabric Adapter

Este tópico descreve como integrar o Elastic Fabric Adapter (EFA) a Podsimplantados no cluster do Amazon EKS. O Elastic Fabric Adapter (EFA) é uma interface de rede para instâncias do Amazon EC2 que permite que você execute aplicações que exigem altos níveis de comunicações entre nós em escala na AWS. Sua interface de hardware de bypass do sistema operacional personalizada melhora a performance das comunicações entre instâncias, o que é essencial para escalar essas aplicações. Com a EFA, as aplicações de Computação de Alta Performance (HPC) que usam a Interface de Passagem de Mensagens (MPI) e as aplicações de Machine Learning (ML) que usam a NVIDIA Collective Communications Library (NCCL) podem ser escaladas para milhares de CPUs ou GPUs. Como resultado, você obtém a performance da aplicação de clusters de HPC on-premises, com a elasticidade e a flexibilidade sob demanda da nuvem AWS. A integração da EFA com aplicações executadas em clusters do Amazon EKS pode reduzir o tempo para concluir workloads de treinamento distribuídas em grande escala sem precisar adicionar instâncias adicionais ao cluster. Para obter mais informações sobre o EFA, consulte Elastic Fabric Adapter.

Tipos de instância com EFA

O plug-in de dispositivo EFA Kubernetes da AWS é compatível com todos os tipos de instância do Amazon EC2 que têm EFA. Para ver uma lista de tipos de instância que oferecem suporte a EFAs, consulte Tipos de instância compatíveis no Guia do usuário do Amazon EC2. No entanto, para executar aplicações de ML rapidamente, recomendamos que uma instância tenha chips de aceleração de hardware, como GPUs nVidia, chips AWS Inferentia ou chips AWS Trainium, além do EFA. Para ver uma lista de tipos de instância equipadas com chips de aceleração de hardware e EFA, consulte Computação acelerada no Guia do usuário do Amazon EC2.

Ao comparar os tipos de instância para escolher entre eles, considere o número de placas de rede EFA disponíveis para esse tipo de instância, bem como o número de placas aceleradoras, a quantidade de CPU e a quantidade de memória. É possível atribuir até uma EFA por placa de rede. Um EFA conta como uma interface de rede. Para ver quantos EFA estão disponíveis para cada tipo de instância equipada com EFA, consulte a lista Placas de rede no Guia do usuário do Amazon EC2.

Pré-requisitos

- Um cluster do existente do Amazon EKS. Se você não tiver um cluster, use um dos nossos manuais de Começar a usar o Amazon EKS para criar um. O cluster deve ser implantado em uma VPC que tenha pelo menos uma sub-rede privada com endereços IP disponíveis suficientes para implantar nós. A sub-rede privada deve ter acesso de saída à Internet fornecido por um dispositivo externo, como um gateway NAT.

  Se você planeja usar o `eksctl` para criar o grupo de nós, o `eksctl` também pode criar um cluster para você.
- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à

versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.

- O Amazon VPC CNI plugin for Kubernetes versão `1.7.10` ou posterior deverá estar instalado para que você possa iniciar os nós de processamento que oferecem suporte a vários Elastic Fabric Adapters, como `p4d` ou `p5`. Para obter mais informações sobre a atualização da sua a versão do Amazon VPC CNI plugin for Kubernetes, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS.

> **Important**
>
> Uma consideração importante, necessária para adotar EFA com o Kubernetes é configurar e gerenciar Huge Pages como um recurso no cluster. Para obter mais informações, consulte Manage Huge Pages (Gerenciar páginas grandes) na documentação do Kubernetes. As instâncias do Amazon EC2 com o driver EFA instalado pré-alocam 5128 páginas enormes de 2 MiB, as quais você pode solicitar como recursos para consumir em suas especificações de trabalho.

Crie grupos de nós.

O procedimento a seguir ajuda você a criar um grupo de nós que conta com o suporte do `p4d.24xlarge` e interfaces EFA e GPUDirect RDMA, além de executar um exemplo de teste NVIDIA Collective Communications Library (NCCL) para a performance do NCCL de vários nós usando EFAs. O exemplo pode ser usado em um modelo para treinamento de aprendizado profundo distribuído no Amazon EKS usando EFAs.

1. Determine quais tipos de instância do Amazon EC2 compatíveis com EFA estão disponíveis na Região da AWS em que você deseja implantar os nós. Substitua *`region-code`* pela Região da AWS em que deseja implantar os grupo de nós.

   ```
   aws ec2 describe-instance-types --region region-code \
       --filters Name=network-info.efa-supported,Values=true \
       --query "InstanceTypes[*].[InstanceType]" --output text
   ```

   Quando você implanta nós, o tipo de instância que você deseja implantar deve estar disponível na Região da AWS em que está o cluster.

2. Determine em quais zonas de disponibilidade o tipo de instância que você deseja implantar está disponível. Neste tutorial, o tipo de instância `p5.48xlarge` é usado e deve ser retornado na saída para o Região da AWS que você especificou na etapa anterior. Ao implantar nós em um cluster de produção, substitua *`p5.48xlarge`* por qualquer tipo de instância retornado na etapa anterior.

```
aws ec2 describe-instance-type-offerings --region region-code \
    --location-type availability-zone --filters Name=instance-
type,Values=p4d.24xlarge,p5.48xlarge \
    --query 'InstanceTypeOfferings[*].Location' --output text
```

Veja um exemplo de saída abaixo.

```
us-west-2a    us-west-2c    us-west-2b
```

Observe as zonas de disponibilidade retornadas para uso em etapas posteriores. Quando você implanta nós em um cluster, a VPC deve ter sub-redes com endereços IP disponíveis em uma das zonas de disponibilidade retornadas na saída.

3. Crie um grupo de nós usando `eksctl`.

Pré-requisito

Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

1. Copie o conteúdo a seguir em um arquivo denominado *`efa-cluster.yaml`*. Substitua *`example values`* pelos seus próprios valores. É possível substituir *`p5.48xlarge`* por uma instância diferente mas, se fizer isso, certifique-se de que os valores de `availabilityZones` sejam as zonas de disponibilidade que foram retornadas para o tipo de instância na etapa 1.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-cluster
  region: region-code
  version: "1.XX"
```

```
iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
```

2. Crie um grupo de nós gerenciados em um cluster existente.

```
eksctl create nodegroup -f efa-cluster.yaml
```

Se você não tiver um cluster existente, recomendamos que siga um dos comandos a seguir para criar um cluster e o grupo de nós.

```
eksctl create cluster -f efa-cluster.yaml
```

> **Note**
>
> Como o tipo de instância usado neste exemplo tem GPUs, o `eksctl` instala automaticamente o plug-in de dispositivo NVIDIA Kubernetes em cada instância para você.

4. Implante o plugin de dispositivo EFA Kubernetes.

O plugin do dispositivo EFA Kubernetes detecta e anuncia interfaces EFA como recursos alocáveis para o Kubernetes. Uma aplicação pode consumir o tipo de recurso estendido `vpc.amazonaws.com/efa` na especificação de uma solicitação de Pod como CPU e memória. Para obter mais informações, consulte Consuming extended resources (Consumir recursos estendidos) na documentação do Kubernetes. Uma vez solicitado, o plug-in atribui e

monta automaticamente uma interface EFA no Pod. Usar o plugin de dispositivo simplifica a configuração do EFA e não requer que um Pod seja executado no modo privilegiado.

```
helm repo add eks https://aws.github.io/eks-charts
helm install aws-efa-k8s-device-plugin --namespace kube-system eks/aws-efa-k8s-device-plugin
```

(Opcional) Testar a performance do EFA

Recomendamos testar a configuração de EFA. Você pode usar os Testes NCCL no repositório `aws-samples/awsome-distributed-training` no GitHub. Os Testes NCCL avaliam a performance da rede usando a Nvidia Collective Communication Library. As etapas a seguir enviam testes de NCCL no Amazon EKS.

1. Implantar o operador Kubeflow MPI

   Para os testes NCCL, você pode aplicar o Kubeflow MPI Operator. O MPI Operator facilita a execução do treinamento distribuído no estilo AllReduce no Kubernetes. Para obter mais informações, consulte MPI Operator no GitHub.
2. Executar o teste de performance NCCL em vários nós para verificar GPUDirectRDMA/EFA

   Para verificar a performance do NCCL com GPUDirectRDMA no EFA, execute o teste de performance padrão da NCCL. Para obter mais informações, consulte o repositório NCCL-Tests (Testes de NCCL) no GitHub.

   Conclua as etapas a seguir para executar um NCCL Performance Test de dois nós. No trabalho de teste de NCCL do exemplo, cada operador solicita oito GPUs, 5210 Mi de `hugepages-2Mi`, quatro EFAs e 8000 Mi de memória, o que significa efetivamente que cada operador consome todos os recursos de uma instância do `p5.48xlarge`.

   a. Criar o manifesto de MPIJob

      Copie o seguinte para um arquivo chamado `nccl-tests.yaml`:

      ```
      apiVersion: kubeflow.org/v2beta1
      kind: MPIJob
      metadata:
        name: nccl-tests
      spec:
      ```

```
  runPolicy:
    cleanPodPolicy: Running
    backoffLimit: 20
  slotsPerWorker: 8
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
         spec:
          restartPolicy: OnFailure
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: test-nccl-launcher
            env:
             - name: PATH
               value: $PATH:/opt/amazon/efa/bin:/usr/bin
             - name: LD_LIBRARY_PATH
               value: /opt/amazon/openmpi/lib:/opt/nccl/build/lib:/opt/amazon/
efa/lib:/opt/aws-ofi-nccl/install/lib:/usr/local/nvidia/lib:$LD_LIBRARY_PATH
             - name: NCCL_DEBUG
               value: INFO
             - name: NCCL_BUFFSIZE
               value: '8388608'
             - name: NCCL_P2P_NET_CHUNKSIZE
               value: '524288'
             - name: NCCL_TUNER_PLUGIN
               value: /opt/aws-ofi-nccl/install/lib/libnccl-ofi-tuner.so
            command:
            - /opt/amazon/openmpi/bin/mpirun
            - --allow-run-as-root
            - --tag-output
            - -np
            - "16"
            - -N
            - "8"
            - --bind-to
            - none
            - -x
            - PATH
            - -x
            - LD_LIBRARY_PATH
            - -x
            - NCCL_DEBUG=INFO
```

```
            - -x
            - NCCL_BUFFSIZE
            - -x
            - NCCL_P2P_NET_CHUNKSIZE
            - -x
            - NCCL_TUNER_PLUGIN
            - --mca
            - pml
            - ^cm,ucx
            - --mca
            - btl
            - tcp,self
            - --mca
            - btl_tcp_if_exclude
            - lo,docker0,veth_def_agent
            - /opt/nccl-tests/build/all_reduce_perf
            - -b
            - "8"
            - -e
            - "16G"
            - -f
            - "2"
            - -g
            - "1"
            - -c
            - "1"
            - -n
            - "100"
    Worker:
      replicas: 2
      template:
        spec:
          nodeSelector:
            node.kubernetes.io/instance-type: "p5.48xlarge"
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: nccl-tests-worker
            volumeMounts:
            - name: shmem
              mountPath: /dev/shm
            resources:
              limits:
                nvidia.com/gpu: 8
```

```
            hugepages-2Mi: 5120Mi
            vpc.amazonaws.com/efa: 32
            memory: 32000Mi
          requests:
            nvidia.com/gpu: 8
            hugepages-2Mi: 5120Mi
            vpc.amazonaws.com/efa: 32
            memory: 32000Mi
      volumes:
      - name: shmem
        hostPath:
          path: /dev/shm
```

b. Aplique o NCCL-tests MPIJob

Envie o `MPIJob` aplicando o manifesto. Isso criará duas instâncias `p5.48xlarge` do Amazon EC2.

```
kubectl apply -f nccl-tests.yaml
```

Veja um exemplo de saída abaixo.

```
mpijob.kubeflow.org/nccl-tests created
```

c. Verifique se o trabalho iniciou os pods

Visualize o Pods em execução.

```
kubectl get pods
```

Veja um exemplo de saída abaixo.

```
NAME                              READY   STATUS     RESTARTS   AGE
nccl-tests-launcher-nbql9    0/1     Init:0/1   0          2m49s
nccl-tests-worker-0          1/1     Running    0          2m49s
nccl-tests-worker-1          1/1     Running    0          2m49s
```

O MPI Operator cria um Pod iniciador e 2 Pods operadores (um em cada nó).

d. Verifique se o trabalho está sendo executado com êxito com os logs

Visualize o log do Pod `nccl-tests-launcher`. Substitua *nbql9* pelo valor do resultado.

```
kubectl logs -f nccl-tests-launcher-nbql9
```

Se o teste for concluído com êxito, você poderá implantar suas aplicações que usam o Nvidia Collective Communication Library.

Implantar workloads de inferência de ML com o AWSInferentia no Amazon EKS

Este tópico descreve como criar um cluster do Amazon EKS com nós que executam as instâncias Amazon EC2 Inf1 e (opcionalmente) implantar uma aplicação de exemplo. As instâncias Inf1 do Amazon EC2 são alimentadas por chips do AWS Inferentia que são personalizados pela AWS para fornecer alta performance e inferência de menor custo na nuvem. Os modelos de machine learning são implantados em contêineres usando o AWS Neuron, um kit de desenvolvimento de software (SDK) especializado que consiste em um compilador, runtime e ferramentas de perfil que otimizam a performance de inferências de machine learning dos chips do AWS Inferentia. O Neuron é compatível com frameworks de machine learning muito usados, como TensorFlow, PyTorch e MXNet.

> **Note**
>
> Os IDs lógicos do dispositivo Neuron devem ser contíguos. Se um Pod que solicita vários dispositivos Neuron estiver agendado em um tipo de instância `inf1.6xlarge` ou `inf1.24xlarge` (que têm mais de um dispositivo Neuron), esse Pod não será iniciado se o agendador do Kubernetes selecionar IDs de dispositivo não contíguos. Para obter mais informações, consulte Device logical IDs must be contiguous (IDS lógicos de dispositivos não devem ser contíguos) no GitHub.

Pré-requisitos

- Ter o `eksctl` instalado no computador. Se ele não estiver instalado, consulte Instalação na documentação do `eksctl`.
- Ter o `kubectl` instalado no computador. Para ter mais informações, consulte Configurar o `kubectl` e o `eksctl`.

- (Opcional) Instale o `python3` no computador. Se você não o tiver instalado, consulte as instruções de instalação de downloads do Python.

Criar um cluster

Para criar um cluster com os nós da instância Inf1 do Amazon EC2

1. Crie um cluster com os nós da instância Inf1 do Amazon EC2. É possível substituir *`inf1.2xlarge`* por qualquer tipo de instância Inf1. O utilitário `eksctl` detecta que você está iniciando um grupo de nós com um tipo de instância `Inf1` e iniciará seus nós utilizando uma das AMIs aceleradas do Amazon Linux otimizadas para Amazon EKS.

   > **Note**
   >
   > Você não pode usar funções do IAM para contas de serviço com o TensorFlow Serving.

   ```
   eksctl create cluster \
       --name inferentia \
       --region region-code \
       --nodegroup-name ng-inf1 \
       --node-type inf1.2xlarge \
       --nodes 2 \
       --nodes-min 1 \
       --nodes-max 4 \
       --ssh-access \
       --ssh-public-key your-key \
       --with-oidc
   ```

   > **Note**
   >
   > Observe o valor da seguinte linha da saída. Ele é usado em uma etapa posterior (opcional).
   >
   > ```
   > [9]  adding identity "arn:aws:iam::111122223333:role/
   > eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09" to auth
   >  ConfigMap
   > ```

Ao iniciar um grupo de nós com instâncias `Inf1`, o `eksctl` instala automaticamente o plug-in de dispositivo AWS Neuron Kubernetes. Este plug-in anuncia dispositivos Neuron como um recurso de sistema para o agendador do Kubernetes, que pode ser solicitado por um contêiner. Além das políticas padrão do IAM do nó do Amazon EKS, a política de acesso somente leitura do Amazon S3 é adicionada para que a aplicação de exemplo, abordada em uma etapa posterior, possa carregar um modelo treinado do Amazon S3.

2. Verifique se todos os Pods foram iniciados corretamente.

```
kubectl get pods -n kube-system
```

Saída abreviada:

```
NAME                                   READY   STATUS    RESTARTS   AGE
[...]
neuron-device-plugin-daemonset-6djhp   1/1     Running   0          5m
neuron-device-plugin-daemonset-hwjsj   1/1     Running   0          5m
```

(Opcional) Implante uma imagem da aplicação do TensorFlow Serving

Um modelo treinado deve ser compilado para um destino do Inferentia antes que possa ser implantado em instâncias do Inferentia. Para continuar, você precisará de um modelo do TensorFlow otimizado para Neuron salvo no Amazon S3. Se você ainda não tem um SavedModel, siga o tutorial para Criar um modelo ResNet50 compatível com Neuron e carregue o SavedModel resultante no S3. O Resnet-50 é um modelo de machine learning popular usado para tarefas de reconhecimento de imagem. Para obter mais informações sobre como compilar modelos do Neuron, consulte The AWS Inferentia Chip With DLAMI no Manual do desenvolvedor do AWS Deep Learning AMI.

O exemplo de manifesto de implantação gerencia um contêiner de serviço de inferência pré-criado para o TensorFlow fornecido pelo AWS Deep Learning Containers. Dentro do recipiente está o AWS Neuron Runtime e a aplicação TensorFlow Serving. Uma lista completa de contêiners de aprendizado profundo criados previamente e otimizados para o Neuron é mantida no GitHub em Available Images (Imagens Disponíveis). Na inicialização, o DLC buscará o modelo do Amazon S3, iniciará o Neuron TensorFlow Serving com o modelo salvo e aguardará as solicitações de previsão.

O número de dispositivos Neuron alocados para a aplicação de serviço pode ser ajustado alterando o recurso `aws.amazon.com/neuron` no yaml de implantação. Observe que a comunicação entre

o TensorFlow Serving e o runtime do Neuron acontece no GRPC, o que requer a transferência do recurso `IPC_LOCK` para o contêiner.

1. Adicione a política `AmazonS3ReadOnlyAccess` do política do IAM à função de instância do nó que foi criada na etapa 1 do Criar um cluster. Isso é necessário para que a aplicação de exemplo possa carregar um modelo treinado do Amazon S3.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
    --role-name eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09
```

2. Crie um arquivo denominado `rn50_deployment.yaml` com o seguinte conteúdo: Atualize o código da região e o caminho do modelo para corresponder às configurações desejadas. O nome do modelo é para fins de identificação quando um cliente faz uma solicitação ao servidor do TensorFlow. Este exemplo usa um nome de modelo para corresponder a um script de cliente ResNet50 de exemplo que será usado em uma etapa posterior para enviar solicitações de previsão.

```
aws ecr list-images --repository-name neuron-rtd --registry-id 790709498068 --
region us-west-2
```

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
    role: master
spec:
  replicas: 2
  selector:
    matchLabels:
      app: eks-neuron-test
      role: master
  template:
    metadata:
      labels:
        app: eks-neuron-test
        role: master
    spec:
      containers:
```

```
        - name: eks-neuron-test
          image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
neuron:1.15.4-neuron-py37-ubuntu18.04
          command:
            - /usr/local/bin/entrypoint.sh
          args:
            - --port=8500
            - --rest_api_port=9000
            - --model_name=resnet50_neuron
            - --model_base_path=s3://your-bucket-of-models/resnet50_neuron/
          ports:
            - containerPort: 8500
            - containerPort: 9000
          imagePullPolicy: IfNotPresent
          env:
            - name: AWS_REGION
              value: "us-east-1"
            - name: S3_USE_HTTPS
              value: "1"
            - name: S3_VERIFY_SSL
              value: "0"
            - name: S3_ENDPOINT
              value: s3.us-east-1.amazonaws.com
            - name: AWS_LOG_LEVEL
              value: "3"
          resources:
            limits:
              cpu: 4
              memory: 4Gi
              aws.amazon.com/neuron: 1
            requests:
              cpu: "1"
              memory: 1Gi
          securityContext:
            capabilities:
              add:
                - IPC_LOCK
```

3. Implante o modelo.

```
kubectl apply -f rn50_deployment.yaml
```

4. Crie um arquivo denominado `rn50_service.yaml` com o conteúdo a seguir. As portas HTTP e gRPC são abertas para aceitar solicitações de previsão.

```
kind: Service
apiVersion: v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
spec:
  type: ClusterIP
  ports:
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: eks-neuron-test
    role: master
```

5. Crie um serviço Kubernetes para a aplicação de fornecimento de modelos TensorFlow.

```
kubectl apply -f rn50_service.yaml
```

(Opcional) Faça previsões em relação ao serviço do TensorFlow Serving

1. Para testar localmente, encaminhe a porta gRPC para o serviço `eks-neuron-test`.

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. Crie um script Python chamado `tensorflow-model-server-infer.py` com o conteúdo a seguir. Esse script executa a inferência via gRPC, que é um framework de serviço.

```
import numpy as np
   import grpc
   import tensorflow as tf
   from tensorflow.keras.preprocessing import image
   from tensorflow.keras.applications.resnet50 import preprocess_input
   from tensorflow_serving.apis import predict_pb2
```

```
   from tensorflow_serving.apis import prediction_service_pb2_grpc
   from tensorflow.keras.applications.resnet50 import decode_predictions

   if __name__ == '__main__':
       channel = grpc.insecure_channel('localhost:8500')
       stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
       img_file = tf.keras.utils.get_file(
           "./kitten_small.jpg",
           "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/
docs/images/kitten_small.jpg")
       img = image.load_img(img_file, target_size=(224, 224))
       img_array = preprocess_input(image.img_to_array(img)[None, ...])
       request = predict_pb2.PredictRequest()
       request.model_spec.name = 'resnet50_inf1'
       request.inputs['input'].CopyFrom(
           tf.make_tensor_proto(img_array, shape=img_array.shape))
       result = stub.Predict(request)
       prediction = tf.make_ndarray(result.outputs['output'])
       print(decode_predictions(prediction))
```

3. Execute o script para enviar previsões ao seu serviço.

```
python3 tensorflow-model-server-infer.py
```

Veja um exemplo de saída abaixo.

```
[[(u'n02123045', u'tabby', 0.68817204), (u'n02127052', u'lynx', 0.12701613),
 (u'n02123159', u'tiger_cat', 0.08736559), (u'n02124075', u'Egyptian_cat',
 0.063844085), (u'n02128757', u'snow_leopard', 0.009240591)]]
```

Organizar e monitorar recursos de clusters

Este capítulo inclui os tópicos a seguir para ajudar você a gerenciar o cluster. Também é possível visualizar informações sobre os recursos do Kubernetes com o AWS Management Console.

- O Kubernetes Dashboard é uma interface de usuário de uso geral baseada na Web para clusters do Kubernetes. Com ele, os usuários podem gerenciar aplicações em execução no cluster, solucionar problemas relacionados a essas aplicações e gerenciar o próprio cluster. Para obter mais informações, consulte o repositório do Kubernetes Dashboard no GitHub.
- Visualizar o uso de recursos com o KubernetesMetrics Server: o servidor de métricas do Kubernetes é um agregador de dados de uso de recursos no cluster. Não é implantado por padrão no cluster, mas é usado por complementos do Kubernetes, como o painel do Kubernetes e o Escalar as implantações de pods com o Horizontal Pod Autoscaler. Neste tópico, você aprenderá como instalar o Metrics Server.
- Implantar aplicações com o Helm no Amazon EKS: o gerenciador de pacotes para o Kubernetes ajuda a instalar e gerenciar aplicações no cluster do Kubernetes. Este tópico ajuda a instalar e executar os binários Helm para que você possa instalar e gerenciar gráficos usando a CLI do Helm no computador local.
- Organizar recursos do Amazon EKS com tags – Para ajudar a gerenciar os recursos do Amazon EKS, você pode atribuir seus próprios metadados a cada recurso na forma de etiquetas. Este tópico descreve as etiquetas e mostra como criá-las.
- Visualizar e gerenciar o Amazon EKS e as cotas de serviço do Fargate – Sua conta da AWS tem cotas padrão, anteriormente chamadas de limites, para cada serviço da AWS. Conheça as cotas do Amazon EKS e saiba como aumentá-las.

Monitorar e otimizar os custos de clusters do Amazon EKS

O monitoramento de custos é um aspecto essencial do gerenciamento dos clusters do Kubernetes no Amazon EKS. Ao obter visibilidade dos custos do cluster, você pode otimizar a utilização de recursos, definir orçamentos e tomar decisões baseadas em dados sobre suas implantações. O Amazon EKS fornece duas soluções de monitoramento de custos, cada uma com suas vantagens exclusivas, para ajudar você a acompanhar e alocar seus custos de forma eficaz:

Dados de alocação de custos divididos de faturamento da AWS para o Amazon EKS: esse recurso nativo se integra perfeitamente ao Console de Faturamento da AWS, permitindo que você analise

e aloque custos usando a mesma interface e fluxos de trabalho familiares que você usa em outros serviços da AWS. Com a alocação de custos dividida, você pode obter insights sobre seus custos do Kubernetes diretamente junto com seus outros gastos com a AWS, o que facilita a otimização holística dos custos em todo o seu ambiente da AWS. Também é possível utilizar os recursos de faturamento da AWS existentes, como Categorias de Custos e Detecção de Anomalias em Custos, para aprimorar ainda mais seus recursos de gerenciamento de custos. Para obter mais informações, consulte Compreender os dados de alocação de custos divididos no Guia do usuário de faturamento da AWS.

Kubecost: o Amazon EKS é compatível com o Kubecost, uma ferramenta de monitoramento de custos do Kubernetes. O Kubecost oferece uma abordagem rica em recursos e nativa do Kubernetes para monitoramento de custos, fornecendo detalhamentos granulares de custos por recursos do Kubernetes, recomendações de otimização de custos e painéis e relatórios prontos para uso. O Kubecost também recupera dados de preços precisos por meio da integração com o Relatório de Custos e Uso da AWS, garantindo que você tenha uma visão precisa dos custos do Amazon EKS. Saiba como instalar o Kubecost.

Visualizar os custos por pod na cobrança da AWS com alocação de custos divididos

Monitoramento de custos usando dados de alocação de custos divididos da AWS para o Amazon EKS

É possível usar dados de alocação de custos divididos da AWS para o Amazon EKS para obter visibilidade granular de custos para seus clusters do Amazon EKS. Isso permite a você analisar, otimizar e estornar o custo e o uso das suas aplicações do Kubernetes. Você aloca os custos da aplicação a unidades de negócios e equipes individuais com base nos recursos de CPU e memória do Amazon EC2 consumidos por sua aplicação do Kubernetes. Os dados de alocação de custos divididos para o Amazon EKS oferecem visibilidade do custo por pod e permitem agregar os dados de custo por pod usando namespace, cluster e outras primitivas do Kubernetes. Veja a seguir exemplos de primitivas do Kubernetes que você pode usar para analisar dados de alocação de custos do Amazon EKS.

- Nome do cluster
- Implantação
- Namespace
- Nó

- Nome da workload
- Tipo de workload

Para obter mais informações sobre como usar dados de alocação de custos divididos, consulte Noções básicas de dados de alocação de custos divididos no Guia do usuário de faturamento da AWS.

Configurar relatórios de custos e utilização

É possível ativar os Dados de alocação de custos divididos para o EKS no console de gerenciamento de custos, na AWS Command Line Interface ou nos AWS SDKs.

Siga estas etapas para usar os Dados de alocação de custos divididos:

1. Opte por Dados de alocação de custos divididos. Para obter mais informações, consulte Habilitar dados de alocação de custos divididos no Guia do usuário do AWS Cost and Usage Report.
2. Inclua os dados em um relatório novo ou existente.
3. Visualize o relatório. É possível usar o console do Gerenciamento de Faturamento e Custos ou visualizar os arquivos do relatório no Amazon Simple Storage Service.

Instalar o Kubecost e acessar o painel

O Amazon EKS oferece suporte ao Kubecost, que você pode usar para monitorar os custos detalhados por recursos do Kubernetes, incluindo Pods, nós, namespaces e rótulos. Como um administrador de plataforma do Kubernetes e líder financeiro, é possível usar o Kubecost para visualizar um detalhamento das cobranças do Amazon EKS, alocar custos e cobrar de forma retroativa unidades organizacionais, como equipes de aplicações. É possível fornecer às suas equipes internas e unidades de negócios dados de custos transparentes e precisos com base em seu faturamento real da AWS. Além disso, você também pode obter recomendações personalizadas para otimização de custos com base no ambiente de infraestrutura e nos padrões de uso em seus clusters. Para obter mais informações sobre o Kubecost, consulte a documentação do Kubecost.

O Amazon EKS fornece um pacote otimizado para a AWS de Kubecost para visibilidade de custos do cluster. É possível usar seus acordos de suporte da AWS existentes para obter suporte.

Instalar o Kubecost usando o Helm

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Começar a usar o Amazon EKS. O cluster deve ter nós do Amazon EC2 porque não é possível executar o Kubecost em nós do Fargate.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.
- A versão 3.9.0 ou posterior do Helm configurada no seu dispositivo ou no AWS CloudShell. Para instalar ou atualizar o Helm, consulte the section called “Implantar aplicações com o Helm”.
- Se o seu cluster for versão `1.23` ou posterior, você deverá ter o the section called “Amazon EBS” instalado no seu cluster.

1. Determine a versão do Kubecost a ser instalada. É possível ver as versões disponíveis em Kubecost/cost-analyzer na Galeria pública do Amazon ECR. Para obter mais informações sobre a compatibilidade das versões do Kubecost e do Amazon EKS, consulte os requisitos de ambiente na documentação do Kubecost.
2. Instale o Kubecost com o seguinte comando. Substitua *`kubecost-version`* pelo valor obtido do ECR, como *`1.108.1`*.

   ```
   helm upgrade -i kubecost oci://public.ecr.aws/kubecost/cost-analyzer --
   version kubecost-version \
       --namespace kubecost --create-namespace \
       -f https://raw.githubusercontent.com/kubecost/cost-analyzer-helm-chart/develop/
   cost-analyzer/values-eks-cost-monitoring.yaml
   ```

   O Kubecost lança novas versões regularmente. É possível atualizar sua versão usando o `helm upgrade`. Por padrão, a instalação inclui um servidor do Prometheus local e `kube-state-metrics`. Você pode personalizar sua implantação para usar o Amazon Managed Service for Prometheus seguindo a documentação em Integrating with Amazon EKS cost monitoring (Integrar com o monitoramento de custos do Amazon EKS). Para obter uma lista de todas as outras definições que você pode configurar, consulte o exemplo de arquivo de configuração no GitHub.

É possível remover o Kubecost do seu cluster com os seguintes comandos.

```
helm uninstall kubecost --namespace kubecost
kubectl delete ns kubecost
```

Instalar o Kubecost usando os complementos do Amazon EKS

Os complementos do Amazon EKS reduzem a complexidade da atualização do Kubecost e do gerenciamento de licenças. Os complementos do EKS são integrados ao AWS Marketplace.

1. Verifique o Kubecost no console do AWS Marketplace e assine.
2. Determine o nome do seu cluster e a região. Verifique se você fez login na AWS CLI com permissões suficientes para gerenciar o EKS.
3. Crie o complemento Kubecost.

```
aws eks create-addon --addon-name kubecost_kubecost --cluster-name
 $YOUR_CLUSTER_NAME --region $AWS_REGION
```

Saiba como remover um complemento do EKS, como o Kubecost.

Acessar o painel do Kubecost

1. Certifique-se de que os Pods necessários estejam em execução.

```
kubectl get pods -n kubecost
```

Veja um exemplo de saída abaixo.

```
NAME                                          READY   STATUS    RESTARTS   AGE
kubecost-cost-analyzer-b9788c99f-5vj5b        2/2     Running   0          3h27m
kubecost-kube-state-metrics-99bb8c55b-bn2br   1/1     Running   0          3h27m
kubecost-prometheus-server-7d9967bfc8-9c8p7   2/2     Running   0          3h27m
```

2. No seu dispositivo, ative o encaminhamento de portas para expor o painel do Kubecost.

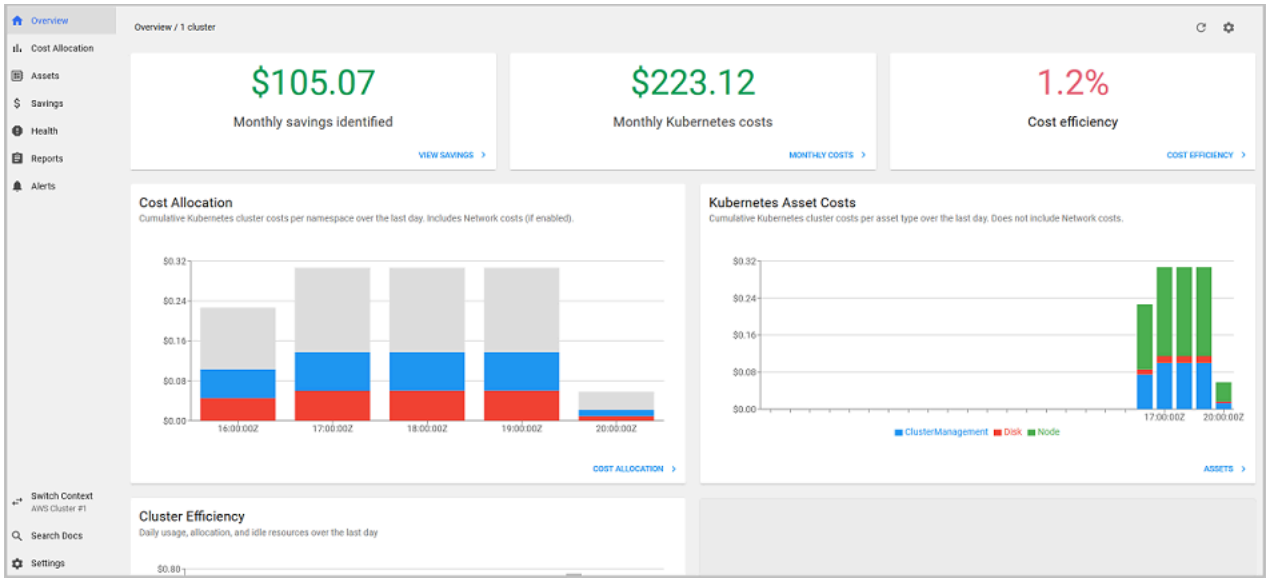
```
kubectl port-forward --namespace kubecost deployment/kubecost-cost-analyzer 9090
```

Como alternativa, é possível usar o AWS Load Balancer Controller para expor o Kubecost e usar o Amazon Cognito para autenticação, autorização e gerenciamento de usuários. Para obter mais informações, consulte Como usar o Application Load Balancer e o Amazon Cognito para autenticar usuários para suas aplicações Kubernetes da Web.

3. No mesmo dispositivo em que você concluiu a etapa anterior, abra um navegador da Web e digite o seguinte endereço.

```
http://localhost:9090
```

Você verá a página de visão geral do Kubecost em seu navegador. Pode levar de 5 a 10 minutos para o Kubecost coletar métricas. É possível ver seus gastos com o Amazon EKS, incluindo custos cumulativos de cluster, custos de ativos do Kubernetes associados e gastos mensais agregados.



4. Para rastrear os custos em um nível de cluster, marque seus recursos do Amazon EKS para cobrança. Para obter mais informações, consulte Marcar recursos para faturamento.

É possível visualizar também as seguintes informações selecionando-as no painel esquerdo do painel:

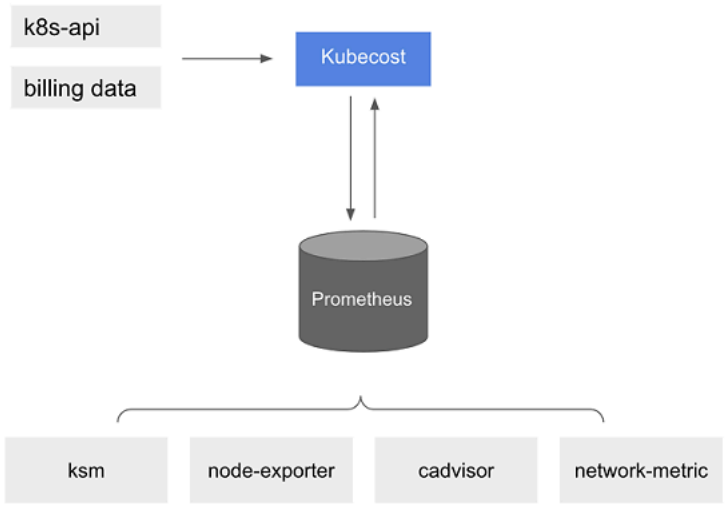
- Cost allocation (Alocação de custos): visualize os custos mensais do Amazon EKS e os custos cumulativos de cada um dos seus namespaces e outras dimensões nos últimos sete dias. Isso é útil para entender quais partes da sua aplicação estão contribuindo para os gastos do Amazon EKS.

- Assets (Ativos): visualize os custos dos ativos de infraestrutura da AWS que estão associados aos recursos do Amazon EKS.

Saiba mais sobre o Kubecost

Recursos adicionais

- Export cost metrics (Exportar métricas de custo): o monitoramento otimizado de custos do Amazon EKS é implantado com o Kubecost e o Prometheus, que é um sistema de monitoramento de código aberto e banco de dados de séries temporais. O Kubecost lê a métrica do Prometheus e, em seguida, realiza cálculos de alocação de custos e grava as métricas novamente no Prometheus. O front-end do Kubecost lê métricas do Prometheus e as mostra na interface do usuário do Kubecost. A arquitetura está ilustrada no diagrama a seguir



  Com o Prometheus pré-instalado, é possível escrever consultas para ingerir dados do Kubecost em seu sistema atual de inteligência de negócios para análise posterior. Também é possível usá-lo como fonte de dados para seu painel Grafana atual para exibir os custos do cluster do Amazon EKS com os quais suas equipes internas estão familiarizadas. Para saber mais sobre como escrever consultas do Prometheus, consulte o arquivo `readme` de Configuração do Prometheus no GitHub ou use os modelos JSON de exemplo do Grafana no repositório Github do Kubecost como referências.

- Integração com o AWS Cost and Usage Report: para realizar cálculos de alocação de custos para o cluster do Amazon EKS, o Kubecost recupera as informações públicas sobre os preços de Serviços da AWS e recursos da AWS da API de tabela de preços da AWS. Também é possível

integrar o Kubecost com o AWS Cost and Usage Report para melhorar a precisão das informações de preços específicas para sua Conta da AWS. Essas informações incluem programas de descontos corporativos, uso de instâncias reservadas, Savings Plans e uso de spot. Para saber mais sobre como a integração do AWS Cost and Usage Report funciona, consulte Integração de faturamento na nuvem AWS na documentação do Kubecost.

Perguntas frequentes

Veja as seguintes perguntas e respostas comuns sobre o uso do Kubecost com o Amazon EKS.

Qual a diferença entre o pacote personalizado do Kubecost e a versão gratuita do Kubecost (também conhecida como OpenCost)?

A AWS e o Kubecost colaboraram para oferecer uma versão personalizada do Kubecost. Esta versão inclui um subconjunto de recursos comerciais sem custo adicional. Consulte a tabela a seguir para ver os recursos incluídos no pacote personalizado do Kubecost.

Atributo	Nível gratuito do Kubecost	Pacote personalizado do Kubecost otimizado para Amazon EKS	Kubecost Enterprise
Implantação	Hospedado pelo usuário	Hospedado pelo usuário	Hospedado pelo usuário ou hospedado pelo Kubecost (SaaS)
Número compatível de clusters	Ilimitado	Ilimitado	Ilimitado
Bancos de dados compatíveis	Prometheus local	Prometheus local ou Amazon Managed Service for Prometheus	Prometheus, Amazon Managed Service for Prometheus, Cortex ou Thanos
Compatibilidade com retenção de banco de dados	15 dias	Dados históricos ilimitados	Dados históricos ilimitados

Atributo	Nível gratuito do Kubecost	Pacote personalizado do Kubecost otimizado para Amazon EKS	Kubecost Enterprise
Retenção de API do Kubecost (ETL)	15 dias	15 dias	Dados históricos ilimitados
Visibilidade dos custos do cluster	Clusters únicos	Vários clusters unificados	Vários clusters unificados
Visibilidade da nuvem híbrida	-	Clusters Amazon EKS e Amazon EKS Anywhere	Compatibilidade com multinuvem e nuvem híbrida
Alertas e relatórios recorrentes	-	Compatibilidade com alertas de eficiência, alertas de orçamento , alertas de alteração de gastos e outros	Compatibilidade com alertas de eficiência, alertas de orçamento , alertas de alteração de gastos e outros
Relatórios salvos	-	Relatórios usando dados de 15 dias	Relatórios usando dados históricos ilimitados
Integração de faturamento na nuvem	Obrigatório para cada cluster individual	Suporte personali zado de preços para AWS (incluind o vários clusters e várias contas)	Suporte personali zado de preços para AWS (incluind o vários clusters e várias contas)
Recomendações de economia	Insights de um único cluster	Insights de um único cluster	Insights de vários clusters
Governança: auditoria s	-	-	Auditoria de eventos históricos de custos
Compatibilidade com autenticação única (SSO)	-	Compatível com Amazon Cognito	Okta, Auth0, PingID, KeyCloak

Atributo	Nível gratuito do Kubecost	Pacote personalizado do Kubecost otimizado para Amazon EKS	Kubecost Enterprise
Regras de controle de acesso com base em função (RBAC) com SAML **2.0**	-	-	Okta, Auth0, PingID, Keycloak
Treinamento e integração corporativos	-	-	Serviço completo de treinamento e integração para FinOps

O que é o recurso de retenção de API do Kubecost (ETL)?

O recurso de ETL do Kubecost agrega e organiza métricas para promover a visibilidade dos custos em vários níveis de granularidade (p. ex., `namespace-level`, `pod-level` e `deployment-level`). Para o pacote personalizado do Kubecost, os clientes obtêm dados e insights das métricas dos últimos 15 dias.

O que é o recurso de alertas e relatórios recorrentes? Quais alertas e relatórios estão incluídos?

Os alertas do Kubecost permitem que as equipes recebam atualizações em tempo real de gastos do Kubernetes, bem como gastos com a nuvem. Os relatórios recorrentes permitem que as equipes recebam visualizações personalizadas do histórico do Kubernetes e de gastos com a nuvem. Ambos são configuráveis usando a UI do Kubecost ou valores Helm. Eles são compatíveis com e-mail, Slack e Microsoft Teams.

O que os relatórios salvos incluem?

Os relatórios salvos do Kubecost são visualizações predefinidas das métricas de custo e eficiência. Eles incluem custo por cluster, namespace, rótulo e muito mais.

O que é a integração de faturamento na nuvem?

A integração com APIs de faturamento do AWS permitem que o Kubecost exiba custos fora do cluster (p. ex., do Amazon S3). Além disso, ela permite que o Kubecost reconcilie as previsões em cluster do Kubecost com dados reais de faturamento para contabilizar o uso de spot, Savings Plans e descontos corporativos.

O que as recomendações de economia incluem?

> O Kubecost fornece insights e automação para ajudar os usuários a otimizar sua infraestrutura e gastos do Kubernetes.

Há alguma cobrança por esta funcionalidade?

Não. Você pode usar esta versão do Kubecost sem custo adicional. Se você quiser mais recursos do Kubecost não incluídos neste pacote, compre uma licença corporativa do Kubecost por meio do AWS Marketplace ou diretamente no Kubecost.

Existe suporte disponível?

Sim. Você pode abrir um caso de suporte com a equipe do AWS Support em Entre em contato com a AWS.

Eu preciso de uma licença para usar recursos do Kubecost fornecidos pela integração com o Amazon EKS?

Nº

É possível integrar o Kubecost com o AWS Cost and Usage Report para obter relatórios mais precisos?

Sim. É possível configurar o Kubecost para ingerir dados do AWS Cost and Usage Report para obter uma visibilidade precisa dos custos, incluindo descontos, preços spot, preços de instâncias reservadas e outros. Para obter mais informações, consulte Integração de faturamento na nuvem AWS na documentação do Kubecost.

Esta versão é compatível com o gerenciamento de custos de clusters do Kubernetes autogerenciados no Amazon EC2?

Não. Esta versão só é compatível com clusters do Amazon EKS.

O Kubecost pode rastrear custos do Amazon EKS no AWS Fargate?

O Kubecost oferece o melhor esforço para mostrar a visibilidade dos custos do cluster para o Amazon EKS no Fargate, mas com menor precisão do que com o Amazon EKS no Amazon EC2. Isso se deve principalmente à diferença na forma como você é cobrado pelo seu uso. Com o Amazon EKS no Fargate, você é cobrado pelos recursos consumidos. Com o Amazon EKS em nós do Amazon EC2, você é cobrado pelos recursos provisionados. O Kubecost calcula o custo de um nó do Amazon EC2 com base na especificação do nó, que inclui CPU, RAM e armazenamento

temporário. Com o Fargate, os custos são calculados com base nos recursos solicitados para os Pods do Fargate.

Como posso obter atualizações e novas versões do Kubecost?

Você pode atualizar sua versão do Kubecost usando procedimentos padrão de atualização do Helm. As versões mais recentes estão na Galeria pública do Amazon ECR.

O **`kubectl-cost`** é compatível com a CLI? Como faço para instalá-lo?

Sim. O `Kubectl-cost` é uma ferramenta de código aberto do Kubecost (Licença Apache 2.0) que fornece acesso da CLI a métricas de alocação de custos do Kubernetes. Para instalar o `kubectl-cost`, consulte Instalação no GitHub.

A interface do usuário do Kubecost é compatível? Como faço para acessá-la?

O Kubecost fornece um painel da Web que você pode acessar por meio do encaminhamento de portas do `kubectl`, de uma entrada ou de um balanceador de carga. Como alternativa, é possível usar o AWS Load Balancer Controller para expor o Kubecost e usar o Amazon Cognito para autenticação, autorização e gerenciamento de usuários. Para obter mais informações, consulte Como usar o Application Load Balancer e o Amazon Cognito para autenticar usuários para suas aplicações da Web do Kubernetes no blog da AWS.

O Amazon EKS Anywhere é compatível?

Nº

Visualizar o uso de recursos com o KubernetesMetrics Server

O Kubernetes Metrics Server é um agregador de dados de uso de recursos no cluster, e não é implantado por padrão em clusters do Amazon EKS. Para obter mais informações, consulte Kubernetes Metrics Server (Configurar o Metrics Server) no GitHub. O Metrics Server, em geral, é usado por outros complementos do Kubernetes, como o Escalar as implantações de pods com o Horizontal Pod Autoscaler ou o Painel do Kubernetes. Para obter mais informações, consulte Resource metrics pipeline (Pipeline de métricas dos recursos) na documentação do Kubernetes. Este tópico explica como implantar o Kubernetes Metrics Server no cluster do Amazon EKS.

> **Important**
>
> As métricas destinam-se à análise pontual e não são uma fonte precisa para análise histórica. Eles não podem ser usados como uma solução de monitoramento ou para outros

fins que não sejam de ajuste de escala automático. Para obter informações sobre as ferramentas de monitoramento, consulte Monitorar a performance de clusters e visualizara logs.

Implantar o Metrics Server

1. Implante o servidor de métricas com o seguinte comando:

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

   Se estiver usando o Fargate, será necessário alterar esse arquivo. Na configuração padrão, o servidor de métricas usa a porta 10250. Essa porta é reservada no Fargate. Substitua as referências à porta 10250 em components.yaml por outra porta, por exemplo, 10251.

2. Verifique se a implantação do `metrics-server` está executando o número desejado de Podscom o comando a seguir.

```
kubectl get deployment metrics-server -n kube-system
```

   Veja um exemplo de saída abaixo.

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           6m
```

3. Teste se o servidor de métricas está funcionando ao exibir o uso de recursos (CPU/memória) dos nós.

```
kubectl top nodes
```

4. Caso receba a mensagem de erro `Error from server (Forbidden)`, você precisará atualizar sua configuração do RBAC do Kubernetes. Sua identidade do RBAC do Kubernetes precisa de permissões suficientes para ler as métricas do cluster. Analise as permissões mínimas exigidas da API do Kubernetes para leitura de métricas no GitHub. Saiba como conceder identidades ao AWS IAM, como perfis, acesso às APIs do Kubernetes.

Implantar aplicações com o Helm no Amazon EKS

O gerenciador de pacotes para o Kubernetes ajuda a instalar e gerenciar aplicações no cluster do Kubernetes. Para obter mais informações, consulte a documentação do Helm. Este tópico ajuda a instalar e executar os binários Helm para que você possa instalar e gerenciar gráficos usando a CLI do Helm no sistema local.

Important

Antes de instalar os gráficos Helm no cluster do Amazon EKS, é necessário configurar o `kubectl` para funcionar com o Amazon EKS. Se você ainda não fez isso, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig antes de continuar. Se o comado a seguir for bem-sucedido para o cluster, a configuração estará correta.

```
kubectl get svc
```

Como instalar os binários Helm no sistema local

1. Execute o comando apropriado para o sistema operacional cliente.
   - Se estiver usando o MacOS com o Homebrew, instale os binários com o comando a seguir.

     ```
     brew install helm
     ```
   - Se estiver usando o Windows com o Chocolatey, instale os binários com o comando a seguir.

     ```
     choco install kubernetes-helm
     ```
   - Se você estiver usando o Linux, instale os binários com os comandos a seguir.

     ```
     curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 >
      get_helm.sh
     chmod 700 get_helm.sh
     ./get_helm.sh
     ```

> **Note**
>
> Se você receber uma mensagem de que o `openssl` deve ser instalado primeiro, poderá instalá-lo com o comando a seguir.
>
> ```
> sudo yum install openssl
> ```

2. Para escolher os novos arquivos binários no PATH, feche a janela atual do terminal e abra uma nova.
3. Veja a versão do Helm que você instalou.

   ```
   helm version | cut -d + -f 1
   ```

   Veja um exemplo de saída abaixo.

   ```
   v3.9.0
   ```

4. Neste ponto, você pode executar quaisquer comandos do Helm (como `helm install` *chart-name*) para instalar, modificar, excluir ou consultar gráficos do Helm no seu cluster. Se é iniciante no Helm e não tem um gráfico específico para instalar, você pode:

   - Experimentar instalando um gráfico de exemplo. Consulte Instalar um gráfico de exemplo no Guia de início rápido do Helm.
   - Crie um gráfico de exemplo e envie-o para o Amazon ECR. Para obter mais informações, consulte os Pushing a Helm chart (Enviar um gráfico Helm) no Manual do usuário do Amazon Elastic Container Registry.
   - Instale um gráfico do Amazon EKS do repositório do GitHub eks-charts ou do ArtifactHub.

Organizar recursos do Amazon EKS com tags

É possível usar tags para ajudar você a gerenciar seus recursos do Amazon EKS. Este tópico fornece uma visão geral da função de etiquetas e mostra como você pode criá-las.

Tópicos

- Conceitos Básicos de Tags

- Marcando seus Recursos
- Restrições de tags
- Marcar recursos para faturamento
- Trabalhando com tags usando o console
- Trabalhar com etiquetas usando a CLI, a API ou o eksctl

> **Note**
>
> Tags são um tipo de metadados que são separados de rótulos e anotações do Kubernetes. Para obter mais informações sobre esses outros tipos de metadados, consulte as seguintes seções na documentação do Kubernetes:
>
> - Etiquetas e seletores
> - Anotações

Conceitos Básicos de Tags

Uma tag um rótulo atribuído a um recurso AWS. Cada tag consiste em uma chave e um valor opcional.

Com tags, é possível categorizar seus recursos da AWS. Por exemplo, você pode categorizar recursos por finalidade, proprietário ou ambiente. Quando você tem muitos recursos do mesmo tipo; é possível identificar rapidamente um recurso específico com base nas tags que você atribuiu a ele. Por exemplo, você pode definir um conjunto de etiquetas para os clusters do Amazon EKS para ajudar a monitorar o proprietário e o nível da pilha de cada cluster. Recomendamos planejar um conjunto consistente de chaves de etiquetas para cada tipo de recurso. Você pode pesquisar e filtrar os recursos de acordo com as tags que adicionar.

Após adicionar uma tag, você pode editar as chaves e os valores das tags ou removê-las de um recurso a qualquer momento. Caso exclua um recurso, todas as respectivas tags também serão excluídas.

As etiquetas não têm significado semântico no Amazon EKS e são interpretadas estritamente como uma sequência de caracteres. É possível definir o valor de uma tag como uma string vazia. No entanto, você não pode definir o valor de uma tag como nulo. Se você adicionar uma etiqueta que tenha a mesma chave de uma etiqueta existente nesse recurso, o novo valor substituirá o antigo.

Se você estiver usando o AWS Identity and Access Management (IAM), poderá controlar quais usuários na sua conta da AWS têm permissão para gerenciar etiquetas.

Marcando seus Recursos

As seguintes etiquetas de suporte para recursos do Amazon EKS:

- clusters
- grupos de nós gerenciados
- Perfis do Fargate

Você pode etiquetar esses recursos usando o seguinte:

- Se você estiver usando o console do Amazon EKS, poderá aplicar tags a recursos novos ou existentes a qualquer momento. Para isso, você pode usar a guia Tags (Etiquetas) na página de recursos relevante. Para ter mais informações, consulte Trabalhando com tags usando o console.
- Se você estiver usando o `eksctl`, poderá aplicar tags a recursos quando eles forem criados usando a opção `--tags`.
- Se você estiver usando a AWS CLI, a API do Amazon EKS ou um AWS SDK, será possível aplicar etiquetas a novos recursos usando o parâmetro `tags` na ação da API relevante. Você também pode aplicar etiquetas aos recursos usando a ação da API `TagResource`. Para mais informações, consulte TagResource.

Ao usar algumas ações de criação de recursos, você também pode especificar tags para o recurso ao mesmo tempo em que o cria. Se as tags não puderem ser aplicadas enquanto um recurso estiver sendo criado, o recurso falhará ao ser criado. Esse mecanismo garante que os recursos que você pretende aplicar tags sejam criados com as etiquetas especificadas por você ou não sejam criados. Se você aplicar tags nos recursos no momento da criação, não precisará executar os scripts de aplicação de tags personalizados após a criação do recurso.

As tags não são propagadas para nenhum outro recurso associado ao recurso criado. Por exemplo, as tags de perfil do Fargate não se propagam para outros recursos associados ao perfil do Fargate, como os Pods que são agendados com ele.

Restrições de tags

As restrições a seguir se aplicam às tags:

- Podem ser associadas no máximo 50 tags a um recurso.
- As chaves de tag não podem ser repetidas para um recurso. As chaves de tag devem ser exclusivas, e cada chave só pode ter um valor.
- As chaves podem ter até 128 caracteres em UTF-8.
- Os valores podem ter até 256 caracteres em UTF-8.
- Se vários Serviços da AWS e recursos usam seu esquema de tags, limite os tipos de caracteres que você usa. Alguns serviços podem ter restrições quanto aos caracteres permitidos. Em geral, os caracteres permitidos são letras, números, espaços e os seguintes caracteres: **+ - = . _ : / @**.
- As chaves e valores das tags diferenciam maiúsculas de minúsculas.
- Não use `aws:`, `AWS:` nem qualquer combinação de letras maiúsculas e minúsculas como um prefixo para chaves ou valores. Esses são reservados para uso pela AWS. Você não pode editar nem excluir chaves nem valores de tags com esse prefixo. As tags com esse prefixo não contam para seu limite de tags por recurso.

Marcar recursos para faturamento

Ao aplicar tags aos clusters do Amazon EKS, é possível pode usá-las para alocação de custos em seus Cost & Usage Reports (Relatórios de custos e uso). Os dados de medição do seu Cost & Usage Reports (Relatório de custos e uso) mostram o uso em todos ps clusters do Amazon EKS. Para obter mais informações, consulte Relatório de custos e uso da AWS no Guia do usuário do AWS Billing.

A tag de alocação de custos gerada pela AWS, especificamente `aws:eks:cluster-name`, permite que você divida os custos de instância do Amazon EC2 pelo cluster individual do Amazon EKS no Explorador de custos. No entanto, essa tag não captura as despesas do ambiente de gerenciamento. A tag é automaticamente adicionada às instâncias do Amazon EC2 que participam de um cluster do Amazon EKS. Esse comportamento acontece independentemente de as instâncias serem provisionadas com o uso de grupos de nós gerenciados do Amazon EKS, com o Karpenter ou diretamente com o Amazon EC2. Essa tag específica não conta para o limite de 50 tags. Para usar a tag, o proprietário da conta deve ativá-la no console do AWS Billing ou usando a API. Quando o proprietário de uma conta de gerenciamento do AWS Organizations ativar a tag, ela também será ativada para todas as contas-membro da organização.

É possível organizar suas informações de faturamento com base nos recursos com os mesmos valores da chave da tag. Por exemplo, é possível aplicar tags a vários recursos com um nome de aplicação específico, e depois organizar suas informações de faturamento. Dessa forma, é possível

ver o custo total daquela aplicação em vários serviços. Para obter mais informações sobre como configurar um relatório de alocação de custos com etiquetas, consulte Relatório mensal de alocação de custos no Guia do usuário do AWS Billing.

> **Note**
>
> Se você tiver acabado de habilitar a criação de relatórios, os dados do mês atual estarão disponíveis para visualização após 24 horas.

O Cost Explorer é uma ferramenta de geração de relatórios que está disponível como parte do nível gratuito da AWS. É possível usar o Cost Explorer para visualizar gráficos dos seus recursos do Amazon EKS dos últimos 13 meses. Também é possível prever o provável valor que você gastará nos próximos três meses. É possível exibir os padrões de quanto você gasta nos recursos da AWS ao longo do tempo. Por exemplo, você pode usar o Cost Explorer para identificar áreas que precisam de uma investigação mais profunda e observar tendências que podem ser usadas para o entendimento dos custos. Também é possível especificar os períodos dos dados e visualizar os dados de tempo por dia ou mês.

Trabalhando com tags usando o console

Usando o console do Amazon EKS, é possível gerenciar as etiquetas associadas aos clusters e grupos de nós gerenciados novos ou existentes.

Quando você selecionar uma página específica do recurso no console do Amazon EKS, a página exibirá uma lista desses recursos. Por exemplo, se você selecionar Clusters no painel de navegação esquerdo, o console exibirá uma lista dos clusters do Amazon EKS. Ao selecionar um recurso de uma dessas listas (por exemplo, um cluster específico), que seja compatível com as etiquetas, será possível vê-las e gerenciá-las na guia Tags (Etiquetas).

Também é possível usar o Editor de etiquetas no AWS Management Console que fornece uma maneira unificada de gerenciar etiquetas. Para obter mais informações, consulte Marcar recursos da AWS com o Tag Editor no Guia do usuário do AWS Tag Editor.

Adição de tags a um recurso na sua criação

Você pode adicionar etiquetas aos clusters, grupos de nós gerenciados e perfis do Fargate do Amazon EKS quando os criar. Para ter mais informações, consulte Criar um cluster do Amazon EKS..

Adição e exclusão de tags a um recurso

É possível adicionar ou excluir tags associadas aos seus clusters diretamente a partir da página do recurso.

Para adicionar ou excluir uma etiqueta em um recurso individual

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Na barra de navegação, selecione a Região da AWS a ser usada.
3. No painel de navegação à esquerda, escolha Clusters.
4. Escolha um cluster específico.
5. Escolha a guia Tags e escolha Gerenciar tags.
6. Na página Manage tags (Gerenciar etiquetas), adicione ou exclua as etiquetas conforme necessário.
   - Para adicionar uma tag, escolha Add tag. Em seguida, especifique a chave e o valor de cada tag.
   - Para remover uma tag, escolha Remove tag (Remover tag).
7. Repita esse processo para cada tag que você deseja adicionar ou excluir.
8. Selecione Update (Atualizar) para concluir.

Trabalhar com etiquetas usando a CLI, a API ou o `eksctl`

Use os seguintes comandos da AWS CLI ou operações da API do Amazon EKS para adicionar, atualizar, listar e remover as etiquetas dos recursos. Somente é possível usar `eksctl` para adicionar etiquetas durante a criação dos novos recursos com um único comando.

Suporte à marcação para recursos do Amazon EKS

Tarefa	AWS CLI	AWS Tools for Windows PowerShell	Ação API
Adicione ou sobrescreva uma ou mais tags.	`tag-resource`	`Add-EKSResourceTag`	`TagResource`

Tarefa	AWS CLI	AWS Tools for Windows PowerShell	Ação API
Exclua uma ou mais tags.	untag-resource	Remove-EKSResourceTag	UntagResource

Os exemplos a seguir mostram como marcar ou desmarcar recursos usando AWS CLI.

Exemplo 1: marcar um cluster existente

O comando a seguir marca um cluster existente.

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

Exemplo 2: desmarcar um cluster existente

O comando a seguir exclui uma etiqueta de um cluster existente.

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Exemplo 3: listar etiquetas para um recurso

O comando a seguir lista as tags que estão associadas a um recurso existente.

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

Ao usar algumas ações de criação de recursos, você também pode especificar tags ao mesmo tempo em que cria o recurso. As ações a seguir são compatíveis com a especificação de tags na criação de um recurso.

Tarefa	AWS CLI	AWS Tools for Windows PowerShell	Ação API	**eksctl**
Criar um cluster	create-cluster	New-EKSCluster	CreateCluster	`create cluster`

Tarefa	AWS CLI	AWS Tools for Windows PowerShell	Ação API	**eksctl**
Crie um grupo de nós gerenciados*	create-nodegroup	New-EKSNodegroup	CreateNodegroup	create nodegroup
Criar um perfil do Fargate	create-fargate-profile	New-EKSFargateProfile	CreateFargateProfile.html	create fargateprofile

* Se você quiser aplicar tags também a instâncias do Amazon EC2 ao criar um grupo de nós gerenciados, crie o grupo de nós gerenciados usando um modelo de execução. Para ter mais informações, consulte Etiquetar instâncias do Amazon EC2. Se as instâncias já existirem, você pode etiquetá-las manualmente. Para obter mais informações, consulte Marcar seus recursos, no Guia do usuário do Amazon EC2.

Visualizar e gerenciar o Amazon EKS e as cotas de serviço do Fargate

O Amazon EKS foi integrado ao Service Quotas, um serviço da AWS que permite visualizar e gerenciar as cotas de um local central. Para obter mais informações, consulte O que são Service Quotas? no Guia do usuário do Service Quotas. Com a integração do Service Quotas, é possível pesquisar rapidamente o valor do Amazon EKS e cotas de serviço do AWS Fargate usando o AWS Management Console e a AWS CLI.

AWS Management Console

Como visualizar as cotas de serviço do Amazon EKS e Fargate usando o AWS Management Console

1. Abra o console Service Quotas em https://console.aws.amazon.com/servicequotas/.
2. No painel de navegação à esquerda, selecione Serviços da AWS.
3. Na lista de Serviços da AWS, procure e selecione Amazon Elastic Kubernetes Service (Amazon EKS) ou AWS Fargate.

Na lista Service quotas (Cotas de serviço) é possível ver o nome da cota de serviço, o valor aplicado (se estiver disponível), a cota da AWS padrão e se o valor da cota for ajustável.

4. Para visualizar informações adicionais sobre uma service quota, como descrição, escolha o nome da cota.
5. (Opcional) Para solicitar um aumento de cota, selecione a cota que deseja aumentar, selecione Solicitar Aumento de Cota, insira ou selecione as informações necessárias e, por fim, selecione Solicitar.

Para trabalhar mais com cotas de serviço usando o AWS Management Console, consulte o Guia do usuário do Service Quotas. Para solicitar o aumento da cota, consulte Requesting a Quota Increase (Solicitar um aumento de cota) no Manual do usuário do Service Quotas.

AWS CLI

Como visualizar as cotas de serviço do Amazon EKS e Fargate usando o AWS CLI

Execute o comando a seguir para visualizar as cotas do Amazon EKS.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code eks \
    --output table
```

Execute o comando a seguir para visualizar as cotas do Fargate.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

> **Note**
>
> A cota retornada é o número de tarefas do Amazon ECS ou de Pods do Amazon EKS que podem ser executados simultaneamente no Fargate nesta conta na Região da AWS atual.

Para trabalhar mais com cotas de serviço usando a AWS CLI, consulte a `service-quotas` na Referência de comandos da AWS CLI. Para solicitar um aumento de quotas, consulte o comando `request-service-quota-increase` na Referência de comandos do AWS CLI.

Cotas de serviço

Nome	Padrão	Ajustá	Descrição
Entradas de acesso por cluster	Cada região compatível: 3.000	Não	O número máximo de entradas de acesso por cluster.
Clusters	Cada região compatível: 100	Sim	O número máximo de clusters do EKS reservados para esta conta na região atual.
Grupos de segurança do ambiente de gerenciamento por cluster	Cada região compatível: 4	Não	O número máximo de grupos de segurança de ambiente de gerenciamento por cluster (são especificados ao criar o cluster).
Assinaturas do EKS Anywhere Enterprise	Cada região compatível: 10	Sim	O número máximo de assinaturas do EKS Anywhere Enterprise nesta conta na região atual.
Perfis do Fargate por cluster	Cada região compatível: 10	Sim	O número máximo de perfis do Fargate por cluster.
Pares de rótulos por seletor de perfil do Fargate	Cada região compatível: 5	Sim	O número máximo de pares de rótulos por

Nome	Padrão	Ajustá	Descrição
			seletor de perfil do Fargate.
Grupos de nós gerenciados por cluster	Cada região compatível: 30	Sim	O número máximo de grupos de nós gerenciados por cluster.
Nós por grupo de nós gerenciados	Cada região compatível: 450	Sim	O número máximo de nós por grupo de nós gerenciados.
Intervalos CIDR de acesso a endpoint público por cluster	Cada região compatível: 40	Não	O número máximo de intervalos CIDR de acesso a endpoint público por cluster (são especificados quando você cria ou atualizada o cluster).
Clusters registrados	Cada região compatível: 10	Sim	O número máximo de clusters registrados nesta conta na região atual.
Seletores por perfil do Fargate	Cada região compatível: 5	Sim	O número máximo de seletores por perfil do Fargate.

> **Note**
>
> Os valores padrão são as cotas iniciais definidas pela AWS. Esses valores padrão são separados do valor real da cota aplicada e das cotas de serviço máximas possíveis. Para obter mais informações, consulte Terminologia do Service Quotas no Guia do usuário do Cotas de serviço.

Essas cotas de serviço estão listadas em Amazon Elastic Kubernetes Service (Amazon EKS) no console de Service Quotas. Para solicitar um aumento de cota para os valores que são mostrados como ajustáveis, consulte Solicitar um aumento de cota no Guia do usuário do Service Quotas.

Cotas de serviço do AWS Fargate

O serviço AWS Fargate está listado nas Service Quotas de algumas listas do console do cotas de serviço. A tabela a seguir descreve apenas as cotas que se aplicam ao Amazon EKS. Também é possível configurar alarmes que alertem você quando o uso se aproximar de uma cota de serviço. Para ter mais informações, consulte Criar um alarme do CloudWatch para monitorar as métricas de uso dos recursos do Fargate.

Novas Contas da AWS podem ter cotas iniciais mais baixas que podem ser aumentadas no decorrer do tempo. O Fargate monitora constantemente o uso da conta em cada Região da AWS e aumenta automaticamente as cotas com base no uso. Também é possível solicitar um aumento de cota para valores que sejam mostrados como ajustáveis. Para obter mais informações, consulte Solicitar um aumento da cota no Guia do usuário do Service Quotas.

Nome	Padrão	Ajustável	Descrição
Contagem de recursos de vCPU sob demanda do Fargate	6	Sim	O número de vCPUs do Fargate que podem ser executadas simultaneamente como sob demanda do Fargate nessa conta, na região atual.

> **Note**
>
> Os valores padrão são as cotas iniciais definidas pela AWS. Esses valores padrão são separados do valor real da cota aplicada e das cotas de serviço máximas possíveis. Para obter mais informações, consulte Terminologia do Service Quotas no Guia do usuário do Service Quotas.

> **Note**
>
> Além disso, o Fargate aplica tarefas do Amazon ECS e cotas de taxa de início de Pods do Amazon EKS. Para obter mais informações, consulte Cotas de controle de utilização AWS Fargate no Guia do Amazon ECS.

Segurança no Amazon EKS

A segurança na nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você contará com um data center e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem – a AWS é responsável por proteger a infraestrutura que executa os serviços da AWS na Nuvem AWS. Para o Amazon EKS, a AWS é responsável pelo ambiente de gerenciamento do Kubernetes, que inclui os nós do ambiente de gerenciamento e o banco de dados `etcd`. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos compliance programsAWS. Para saber mais sobre os programas de conformidade que se aplicam ao Amazon EKS, consulte Serviços da AWS no escopo pelo programa de conformidade.
- Segurança na nuvem: sua responsabilidade inclui as seguintes áreas:
  - A configuração de segurança do plano de dados, incluindo a configuração dos grupos de segurança que permitem que o tráfego passe do plano de controle do Amazon EKS para a VPC do cliente
  - A configuração dos nós e dos contêineres
  - O sistema operacional do nó (incluindo atualizações e patches de segurança)
  - Outros softwares de aplicações associadas:
    - Configurar e gerenciar os controles de rede, como regras de firewall
    - Administrar o gerenciamento de acesso e identidade no nível da plataforma, com ou além do IAM.
  - A confidencialidade dos dados, os requisitos da empresa e as leis e regulamentos aplicáveis

Esta documentação ajuda a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon EKS. Os tópicos a seguir mostram como configurar o Amazon EKS para atender aos seus objetivos de segurança e de conformidade. Saiba também como usar outros serviços da AWS que ajudam você a monitorar e proteger os recursos do Amazon EKS.

> **Note**
>
> Os contêineres Linux são compostos de grupos de controle (cgroups) e namespaces que ajudam a limitar o que um contêiner pode acessar, mas todos os contêineres compartilham o mesmo kernel Linux que a instância host do Amazon EC2. Executar um contêiner como usuário raiz (UID 0) ou conceder acesso a um contêiner aos recursos do host ou namespaces, como a rede host ou o namespace PID do host são fortemente desencorajados, pois isso reduz a eficácia do isolamento fornecido pelos contêineres.

Tópicos

Assinatura de certificado

A API de certificados do Kubernetes automatiza o provisionamento de credenciais X.509. A API conta com uma interface de linha de comando para que os clientes de API do Kubernetes solicitem e recebam certificados X.509 de uma autoridade de certificação (CA). Você pode usa o recurso `CertificateSigningRequest` (CSR) para solicitar que um signatário indicado assine o certificado. As solicitações são aprovadas ou negadas antes de serem assinadas. O Kubernetes é compatível com os signatários integrados e signatários personalizados com comportamentos bem definidos. Dessa maneira, os clientes podem prever o que acontece com suas CSRs. Para saber mais sobre assinatura de certificados, consulte solicitações de assinatura.

Um dos signatários integrados é `kubernetes.io/legacy-unknown`. A API `v1beta1` do recurso CSR respeitou esse signatário legacy-unknown. Porém, a API estável `v1` de CSR não permite que o `signerName` seja definido como `kubernetes.io/legacy-unknown`.

A versão `1.21` e anteriores do Amazon EKS permitiam o valor `legacy-unknown` como o `signerName` na API de CSR `v1beta1`. Essa API permite que a Autoridade de certificação (CA) do Amazon EKS gere certificados. Porém, no Kubernetes versão `1.22`, a API do CSR `v1beta1` foi substituída pela API do CSR `v1`. Essa API não oferece suporte ao signerName “legacy-unknown”. Se você desejar usar a CA do Amazon EKS para gerar certificados em seus clusters, deverá usar um signatário personalizado. Ele foi introduzido na versão `1.22` do Amazon EKS. Para usar a versão `v1` da API de CSR e gerar um novo certificado, você deverá migrar todos os manifestos e clientes de API existentes. Certificados existentes criados com a API `v1beta1` existente são válidos e funcionarão até o certificado expirar. Essa transmissão inclui o seguinte:

- Distribuição de confiança: nada. Não há confiança ou distribuição padrão para esse signatário em um cluster do Kubernetes.
- Requerentes permitidos: qualquer
- Extensões x509 permitidas: respeita subjectAltName e extensões de uso de chaves e descarta outras extensões
- Usos de chaves permitidos: não deve incluir usos além de ["criptografia de chaves”, “assinatura digital”, “autenticação de servidor"]

  > **Note**
  >
  > A assinatura de certificados de clientes não é possível.

- Vida útil de expiração/certificado: um ano (padrão e máximo)
- Bit de CA permitido/proibido: não permitido

Exemplo de geração de CSR com signerName

Estas etapas mostram como gerar um certificado de serviço para o nome de DNS `myserver.default.svc` usando `signerName: beta.eks.amazonaws.com/app-serving`. Use isso como orientação para seu próprio ambiente.

1. Execute o comando `openssl genrsa -out myserver.key 2048` para gerar uma chave RSA privada.

```
openssl genrsa -out myserver.key 2048
```

2. Use o seguinte comando para gerar uma solicitação de certificado:

```
openssl req -new -key myserver.key -out myserver.csr -subj "/
CN=myserver.default.svc"
```

3. Gere um valor em `base64` para a solicitação de CSR e armazene-o em uma variável para uso em uma etapa posterior.

```
base_64=$(cat myserver.csr | base64 -w 0 | tr -d "\n")
```

4. Execute o seguinte comando para criar um arquivo chamado `mycsr.yaml`. No exemplo a seguir, `beta.eks.amazonaws.com/app-serving` é `signerName`.

```
cat >mycsr.yaml <<EOF
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: $base_64
  signerName: beta.eks.amazonaws.com/app-serving
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

5. Envie a CSR.

```
kubectl apply -f mycsr.yaml
```

6. Aprove o certificado de serviço.

```
kubectl certificate approve myserver
```

7. Confira se o certificado foi emitido.

```
kubectl get csr myserver
```

Veja um exemplo de saída abaixo.

```
NAME       AGE     SIGNERNAME                              REQUESTOR
 CONDITION
myserver   3m20s   beta.eks.amazonaws.com/app-serving   kubernetes-admin
 Approved,Issued
```

8. Exporte o certificado emitido.

```
kubectl get csr myserver -o jsonpath='{.status.certificate}'| base64 -d
 > myserver.crt
```

Considerações sobre a assinatura de certificado antes de atualizar seu cluster para o Kubernetes 1.24

No Kubernetes `1.23` e anteriores, certificados servidos pelo `kubelet` com Subject Alternative Names (SANs) IP e DNS não verificáveis são emitidos automaticamente com SANs não verificáveis. Os SANs são omitidos no certificado provisionado. Em clusters `1.24` e posteriores, os certificados servidos pelo `kubelet` não são emitidos se o SAN não puder ser verificado. Isso impede o funcionamento dos comandos `kubectl exec` e `kubectl logs`.

Antes de atualizar seu cluster para `1.24`, execute as seguintes etapas a fim de determinar se ele tem certificate signing requests (CSR – Solicitações de assinatura de certificado) que não foram aprovadas:

1. Execute o seguinte comando .

```
kubectl get csr -A
```

Veja um exemplo de saída abaixo.

```
NAME        AGE   SIGNERNAME                        REQUESTOR
                        REQUESTEDDURATION   CONDITION
csr-7znmf   90m   kubernetes.io/kubelet-serving
 system:node:ip-192-168-42-149.region.compute.internal      <none>
 Approved
```

```
csr-9xx5q   90m   kubernetes.io/kubelet-serving
 system:node:ip-192-168-65-38.region.compute.internal      <none>
 Approved, Issued
```

Se a saída retornada mostrar uma CSR com um signatário `kubernetes.io/kubelet-serving` que esteja `Approved` mas não `Issued` para um nó, você precisará aprovar a solicitação.

2. Aprove a CSR manualmente. Substitua `csr-7znmf` pelos seus próprios valores.

```
kubectl certificate approve csr-7znmf
```

Para aprovar automaticamente as CSRs no futuro, recomendamos que você crie um controlador de aprovação capaz de validar e aprovar automaticamente as CSRs que contenham SANs IP ou DNS que o Amazon EKS não possa verificar.

Identity and Access Management para o Amazon EKS

AWS Identity and Access Management (IAM) é um serviço da Serviço da AWS que ajuda o administrador no controle de segurança de acesso aos recursos da AWS de forma segura. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para usar os recursos do Amazon EKS. O IAM é um Serviço da AWS que pode ser usado sem custo adicional.

Público

A forma de usar o AWS Identity and Access Management (IAM) varia em função do trabalho realizado no Amazon EKS

Usuário do serviço: se você usar o serviço Amazon EKS para fazer sua tarefa, o administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais recursos do Amazon EKS para realizar o trabalho, talvez sejam necessárias permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao seu administrador. Se você não puder acessar um recurso na Amazon EKS, consulte Solução de problemas do IAM.

Administrador do serviço: se você for o responsável pelos recursos do Amazon EKS em sua empresa, você provavelmente terá acesso total ao Amazon EKS. Cabe a você determinar quais funcionalidades e recursos do Amazon EKS os usuários do seu serviço devem acessar. Assim,

você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como a empresa pode usar o IAM com o Amazon EKS, consulte Como o Amazon EKS funciona com o IAM.

Administrador do IAM: se você for um administrador do IAM, talvez queira saber detalhes sobre como pode escrever políticas para gerenciar o acesso à Amazon EKS. Para visualizar exemplos de políticas baseadas em identidade do Amazon EKS que podem ser usadas no IAM, consulte Exemplos de políticas baseadas em identidade do Amazon EKS.

Autenticando com identidades

A autenticação é a forma como você faz login na AWS usando suas credenciais de identidade. É necessário ser autenticado (fazer login na AWS) como Usuário raiz da conta da AWS, como usuário do IAM, ou assumindo um perfil do IAM.

Você pode fazer login na AWS como uma identidade federada usando credenciais fornecidas por uma fonte de identidades. AWS IAM Identity Center Os usuários (IAM Identity Center), a autenticação única da empresa e as suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Quando você acessa a AWS usando a federação, está indiretamente assumindo um perfil.

A depender do tipo de usuário, você pode fazer login no AWS Management Console ou no portal de acesso AWS. Para obter mais informações sobre como fazer login na AWS, consulte Como fazer login na contaConta da AWS no Início de Sessão da AWS Guia do usuário .

Se você acessar a AWS programaticamente, a AWS fornecerá um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para você assinar criptograficamente as solicitações usando as suas credenciais. Se você não utilizar as ferramentas AWS, deverá designar as solicitações por conta própria. Para obter mais informações sobre como usar o método recomendado para designar solicitações por conta própria, consulte Designando solicitações de API AWS no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, a AWS recomenda o uso da autenticação multifator (MFA) para aumentar a segurança de sua conta. Para saber mais, consulte Autenticação Multifator no Guia do Usuário do AWS IAM Identity Center. Usar a autenticação multifator (MFA) na AWS no Guia do Usuário do IAM.

Usuário raiz Conta da AWS

Ao criar uma Conta da AWS, você começa com uma identidade de login com acesso completo a todos os Serviços da AWS e recursos na conta. Essa identidade, chamada usuário raiz da Conta da AWS, é acessada por login com o endereço de e-mail e a senha usada para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do Usuário do IAM.

Usuários e grupos do IAM

Um usuário do IAM é uma identidade dentro da Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do Usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de um perfil) no Guia do usuário do IAM.

Perfis do IAM

Um perfil do IAM é uma identidade dentro da Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. É possível presumir temporariamente um perfil do IAM no AWS Management Console alternando perfis. É possível presumir um perfil chamando uma operação de API da AWS CLI ou da AWS, ou usando

um URL personalizado. Para obter mais informações sobre métodos para o uso de perfis, consulte Utilizar perfis do IAM no Guia do usuário do IAM.

Funções do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidades de terceiros no Guia do Usuário do IAM. Se você usar o Centro de identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no Manual do Usuário do AWS IAM Identity Center.
- Permissões temporárias para usuários do IAM –- um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas –- é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, alguns Serviços da AWS permitem que você anexe uma política diretamente a um recurso (em vez de usar um perfil como proxy). Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.
- Acesso entre serviços: alguns Serviços da AWS usam atributos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.
  - Encaminhamento de sessões de acesso (FAS): qualquer pessoa que utilizar uma função ou usuário do IAM para realizar ações na AWS é considerada uma entidade principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O recurso FAS utiliza as permissões da entidade principal que chama um Serviço da AWS, combinadas às permissões do Serviço da AWS solicitante, para realizar solicitações para serviços downstream. As solicitações de FAS só são feitas quando um serviço recebe uma solicitação que exige interações com outros Serviços da AWS ou com recursos para serem concluídas. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.

  - Função de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um Serviço da AWS no Guia do Usuário do IAM.
  - Perfil vinculado a serviço: um perfil vinculado a serviço é um tipo de perfil de serviço vinculado a um Serviço da AWS. O serviço pode presumir a função de executar uma ação em seu nome. Funções vinculadas ao serviço aparecem em sua Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para funções vinculadas ao serviço.
- Aplicações em execução no Amazon EC2: é possível usar um perfil do IAM para gerenciar credenciais temporárias para aplicações em execução em uma instância do EC2 e fazer solicitações da AWS CLI ou da AWS API. É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir um perfil da AWS a uma instância do EC2 e disponibilizá-la para todas as suas aplicações, crie um perfil de instância que esteja anexado a ela. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Utilizar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar perfis do IAM, consulte Quando criar um perfil do IAM (em vez de um usuário) no Guia do usuário do IAM.

Gerenciando acesso usando políticas

Você controla o acesso na AWS criando políticas e anexando-as a identidades ou atributos da AWS. Uma política é um objeto na AWS que, quando associado a uma identidade ou recurso, define suas permissões. A AWS avalia essas políticas quando uma entidade principal (usuário, usuário raiz ou sessão de perfil) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada na AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do Usuário do IAM.

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a o quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissões para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do

IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem presumir os perfis.

As políticas do IAM definem permissões para uma ação independente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de perfis do AWS Management Console, da AWS CLI ou da API da AWS.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas independentes que podem ser anexadas a vários usuários, grupos e perfis na Conta da AWS. As políticas gerenciadas incluem políticas gerenciadas pela AWS e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do Usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. As entidades principais podem incluir contas, usuários, perfis, usuários federados ou Serviços da AWS.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Não é possível usar as políticas gerenciadas da AWS do IAM em uma política baseada em atributos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes as políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Amazon S3, AWS WAF e Amazon VPC são exemplos de serviços que oferecem compatibilidade com ACLs. Para saber mais sobre ACLs, consulte Visão geral da lista de controle de acesso (ACL) no Guia do Desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

A AWS oferece compatibilidade com tipos de política menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um atributo avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do Usuário do IAM.
- Políticas de controle de serviço (SCPs) –- SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (UO) no AWS Organizations. O AWS Organizations é um serviço que agrupa e gerencia centralmente várias Contas da AWS pertencentes a sua empresa. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades em contas membro, o que inclui cada Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizations e SCPs, consulte Service control policies no Guia do usuário do AWS Organizations.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas

substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do Usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como a AWS determina permitir ou não uma solicitação quando há vários tipos de política envolvidos, consulte Lógica da avaliação de políticas no Guia do Usuário do IAM.

Como o Amazon EKS funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon EKS, você deve entender quais recursos do IAM estão disponíveis para uso com o Amazon EKS. Para obter uma visualização de alto nível de como o Amazon EKS e outros serviços da AWS funcionam com o IAM, consulte Serviços da AWS compatíveis com o IAM no Manual do usuário do IAM.

Tópicos

- Políticas baseadas em identidade do Amazon EKS
- Políticas baseadas em recursos do Amazon EKS
- Autorização baseada em etiquetas do Amazon EKS
- Funções do IAM no Amazon EKS

Políticas baseadas em identidade do Amazon EKS

Com as políticas baseadas em identidade do IAM, é possível especificar ações ou recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. A Amazon EKS oferece suporte a ações, chaves de condição e recursos específicos. Para saber mais sobre todos os elementos usados em uma política JSON, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

Ações

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a o quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Action` de uma política JSON descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de políticas geralmente têm o mesmo nome que a operação

de API da AWS associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

As ações de política na Amazon EKS usam o seguinte prefixo antes da ação: `eks:`. Por exemplo, para conceder a alguém permissão para obter informações descritivas sobre um cluster do Amazon EKS, inclua a ação `DescribeCluster` na política dessa pessoa. As instruções de política devem incluir um elemento `Action` ou `NotAction`.

Para especificar várias ações em uma única instrução, separe-as com vírgulas, como segue:

```
"Action": ["eks:action1", "eks:action2"]
```

Você também pode especificar várias ações usando caracteres curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "eks:Describe*"
```

Para ver uma lista das ações do Amazon EKS, consulte Ações definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço.

Recursos

Os administradores podem usar AWS as políticas JSON para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

O recurso de cluster do Amazon EKS tem o seguinte ARN.

```
arn:aws:eks:region-code:account-id:cluster/cluster-name
```

Para obter mais informações sobre o formato de ARNs, consulte Nomes de recursos da Amazon (ARNs) e namespaces de serviços da AWS.

Por exemplo, para especificar o cluster com o nome *my-cluster* em sua instrução, use o seguinte ARN:

```
"Resource": "arn:aws:eks:region-code:111122223333:cluster/my-cluster"
```

Para especificar todos os clusters que pertencem a uma conta e Região da AWS específicas, use o caractere curinga (*):

```
"Resource": "arn:aws:eks:region-code:111122223333:cluster/*"
```

Algumas ações do Amazon EKS, como as que servem para a criação de recursos, não podem ser executadas em um recurso específico. Nesses casos, você deve utilizar o caractere curinga (*).

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do Amazon EKS e seus ARNs, consulte Recursos definidos pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo Amazon Elastic Kubernetes Service.

Chaves de condição

O Amazon EKS define seu próprio conjunto de chaves de condição e também oferece suporte ao uso de algumas chaves de condição globais. Para ver todas as chaves de condição globais da AWS, consulte Chaves de contexto de condição globais da AWS no Guia do usuário do IAM.

Você pode definir chaves de condição ao associar um provedor OpenID Connect ao cluster. Para ter mais informações, consulte Exemplo de política do IAM.

Todas as ações do Amazon EC2 oferecem suporte às chaves de condição `aws:RequestedRegion` e `ec2:Region`. Para obter mais informações, consulte Exemplo: restrição de acesso a uma Região da AWSespecífica.

Para obter uma lista de chaves de condição do Amazon EKS, consulte Condições definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar a chave de condição, consulte Ações definidas pelo Amazon Elastic Kubernetes Service.

Exemplos

Para visualizar exemplos de políticas baseadas em identidade do Amazon EKS, consulte Exemplos de políticas baseadas em identidade do Amazon EKS.

Quando você cria um cluster do Amazon EKS, a entidade principal do IAM que cria o cluster, recebe automaticamente permissões `system:masters` na configuração de controle de acesso baseado em perfil (RBAC) no ambiente de gerenciamento do Amazon EKS. Como essa entidade principal não é exibida em nenhuma configuração visível, mantenha o controle de qual entidade principal criou o cluster originalmente. Para conceder a outras entidades principais do IAM a capacidade de interagir com o cluster, edite o `aws-auth ConfigMap` no Kubernetes e crie um Kubernetes `rolebinding` ou `clusterrolebinding` com o nome de um `group` que você especifica no `aws-auth ConfigMap`.

Para obter mais informações sobre como trabalhar com o ConfigMap, consulte Conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes.

Políticas baseadas em recursos do Amazon EKS

O Amazon EKS não oferece suporte a políticas baseadas em recursos.

Autorização baseada em etiquetas do Amazon EKS

É possível anexar etiquetas aos recursos do Amazon EKS ou passar etiquetas em uma solicitação para o Amazon EKS. Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`. Para obter mais informações sobre recursos de marcação do Amazon EKS, consulte Organizar recursos do Amazon EKS com tags. Para obter mais informações sobre com quais ações você pode usar tags em chaves de condição, consulte Actions defined by Amazon EKS (Ações definidas pelo Amazon EKS) na Referência de autorização do serviço.

Funções do IAM no Amazon EKS

Perfil do IAM é uma entidade dentro da sua conta da AWS que tem permissões específicas.

Usar credenciais temporárias com o Amazon EKS

É possível usar credenciais temporárias para fazer login com federação, assumir um perfil do IAM ou assumir um perfil entre contas. As credenciais de segurança temporárias são obtidas chamando operações da API do AWS STS, como AssumeRole ou GetFederationToken.

A Amazon EKS oferece suporte ao uso de credenciais temporárias.

Funções vinculadas a serviço

Perfis vinculados ao serviço permitem que os serviços da AWS acessem recursos em outros serviços para concluir uma ação em seu nome. Os perfis vinculados a serviço aparecem em sua conta do IAM e são de propriedade do serviço. Um administrador pode visualizar, mas não pode editar as permissões de funções vinculadas ao serviço.

Amazon EKS oferece suporte às funções de serviço. Para obter detalhes sobre como criar ou gerenciar funções vinculadas ao serviço do Amazon EKS, consulte Usar funções vinculadas ao serviço para o Amazon EKS.

Perfis de serviço

Esse atributo permite que um serviço assuma um perfil de serviço em seu nome. O perfil permite que o serviço acesse recursos em outros serviços para concluir uma ação em seu nome. Os perfis de serviço aparecem em sua conta do IAM e são de propriedade da conta. Isso significa que um administrador do IAM pode alterar as permissões para esse perfil. Porém, fazer isso pode alterar a funcionalidade do serviço.

Amazon EKS oferece suporte às funções de serviço. Para ter mais informações, consulte Função do IAM do cluster do Amazon EKS e Perfil do IAM em nós do Amazon EKS.

Escolher uma função IAM no Amazon EKS

Ao criar um recurso de cluster no Amazon EKS, é necessário escolher uma função para permitir que o Amazon EKS acesse outros recursos da AWS em seu nome. Se você já tiver criado uma função de serviço, o Amazon EKS fornecerá uma lista de funções da qual escolher. É importante escolher uma função que tenha as políticas gerenciadas do Amazon EKS anexadas. Para ter mais informações, consulte Verificar se há uma função de cluster existente e Verificar se há uma função existente do nó.

Exemplos de políticas baseadas em identidade do Amazon EKS

Por padrão, os usuários e as funções do IAM não têm permissão para criar ou modificar recursos do Amazon EKS. Eles também não podem executar tarefas usando o AWS Management Console, a AWS CLI ou uma API da AWS. Um administrador do IAM deve criar políticas do IAM que concedam aos usuários e perfis permissão para executarem operações de API específicas nos recursos especificados de que precisam. O administrador deve anexar essas políticas aos usuários ou grupos do IAM que exigem essas permissões.

Para saber como criar uma política baseada em identidade do IAM utilizando esses exemplos de documentos de política JSON, consulte Criar políticas na guia JSON no Guia do usuário do IAM.

Quando você cria um cluster do Amazon EKS, a entidade principal do IAM que cria o cluster, recebe automaticamente permissões `system:masters` na configuração de controle de acesso baseado em perfil (RBAC) no ambiente de gerenciamento do Amazon EKS. Como essa entidade principal não é exibida em nenhuma configuração visível, mantenha o controle de qual entidade principal criou o cluster originalmente. Para conceder a outras entidades principais do IAM a capacidade de interagir com o cluster, edite o `aws-auth ConfigMap` no Kubernetes e crie um Kubernetes `rolebinding` ou `clusterrolebinding` com o nome de um `group` que você especifica no `aws-auth ConfigMap`.

Para obter mais informações sobre como trabalhar com o ConfigMap, consulte Conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes.

Tópicos

- Melhores práticas de política
- Usar o console do Amazon EKS
- Permitir que os usuários do IAM visualizem suas próprias permissões
- Criar um cluster do Kubernetes na Nuvem AWS
- Criar um cluster local do Kubernetes em um Outpost
- Atualizar um cluster do Kubernetes
- Listar ou descrever todos os clusters

Melhores práticas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon EKS em sua conta. Essas ações podem incorrer em custos para a Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas gerenciadas pela AWS e avance para as permissões de privilégio mínimo: para começar a conceder permissões a seus usuários e workloads, use as políticas gerenciadas pela AWS que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis na sua Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da AWS específicas para seus casos de uso. Para mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso a ações de serviço, se elas forem usadas por meio de um Serviço da AWS específico, como o AWS CloudFormation. Para obter mais informações, consulte Elementos de política JSON do IAM: Condition no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações acionáveis para ajudar você a criar políticas seguras e funcionais. Para mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA): se houver um cenário que exija usuários do IAM ou um usuário raiz em sua Conta da AWS, ative a MFA para obter segurança adicional. Para exigir a MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do usuário do IAM.

Para mais informações sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usar o console do Amazon EKS

Para acessar o console da Amazon EKS, uma entidade principal do IAM deve ter um conjunto mínimo de permissões. Essas permissões devem garantir à entidade principal concessão para visualizar os detalhes sobre os recursos da Amazon EKS na sua conta da AWS. Se você criar uma política baseada em identidade que seja mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para as entidades principais com essa política anexada.

Para garantir que as entidades principais do IAM ainda possam usar o console do Amazon EKS, crie uma política com seu próprio nome exclusivo, como `AmazonEKSAdminPolicy`. Anexe a política às entidades principais. Para obter mais informações, consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do IAM.

> **Important**
>
> A política de exemplo a seguir permite que uma entidade principal visualize informações na guia Configuração do console. Para visualizar informações nas guias Visão geral e Recursos no AWS Management Console, a entidade principal também precisa de permissões do Kubernetes. Para ter mais informações, consulte Permissões obrigatórias.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
```

```
                }
            }
        }
    ]
}
```

Não é necessário conceder permissões mínimas do console para perfis que fazem chamadas somente à AWS CLI ou à API da AWS. Em vez disso, permita o acesso somente às ações que corresponderem a operação da API que você estiver tentando executar.

Permitir que os usuários do IAM visualizem suas próprias permissões

Este exemplo mostra como você pode criar uma política que permite que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou de forma programática usando a AWS CLI ou a AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
```

```
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Criar um cluster do Kubernetes na Nuvem AWS

Este exemplo de política inclui as permissões mínimas necessárias para criar um cluster do Amazon EKS chamado *my-cluster* na Região da AWS *us-west-2*. Você pode substituir a Região da AWS pela Região da AWS na qual você deseja criar o cluster. Se você receber um aviso informando que As ações na sua política não são compatíveis com permissões em nível de recurso e exigem que você escolha **`All resources`** no AWS Management Console, é possível ignorá-lo com segurança. Se sua conta já tiver o perfil *AWSServiceRoleForAmazonEKS*, você poderá remover a ação `iam:CreateServiceLinkedRole` da política. Se você já tiver criado um cluster do Amazon EKS em sua conta, esse perfil já existirá, a menos que você o tenha excluído.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
eks.amazonaws.com/AWSServiceRoleForAmazonEKS",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iam:AWSServiceName": "eks"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        }
```

```
    ]
}
```

Criar um cluster local do Kubernetes em um Outpost

Este exemplo de política inclui as permissões mínimas necessárias para criar um cluster local do Amazon EKS denominado *my-cluster* em um Outpost na Região da AWS *us-west-2* . Você pode substituir a Região da AWS pela Região da AWS na qual você deseja criar o cluster. Se você receber um aviso informando que As ações na sua política não são compatíveis com permissões em nível de recurso e exigem que você escolha **`All resources`** no AWS Management Console, é possível ignorá-lo com segurança. Se sua conta já tiver o perfil `AWSServiceRoleForAmazonEKSLocalOutpost`, você poderá remover a ação `iam:CreateServiceLinkedRole` da política. Se você já tiver criado um cluster local do Amazon EKS na sua conta, esse perfil já existirá, a menos que você o tenha excluído.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "iam:GetRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/outposts.eks-
local.amazonaws.com/AWSServiceRoleForAmazonEKSLocalOutpost"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
```

```
                "iam:ListAttachedRolePolicies"
            ]
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": "arn:aws:iam::*:instance-profile/eks-local-*",
            "Effect": "Allow"
        },
    ]
}
```

Atualizar um cluster do Kubernetes

Este exemplo de política inclui a permissão mínima necessária para atualizar um cluster chamado *my-cluster* na Região da AWS us-west-2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:UpdateClusterVersion",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        }
    ]
}
```

Listar ou descrever todos os clusters

Este exemplo de política inclui as permissões mínimas necessárias para listar e descrever todos os clusters na sua conta. Uma entidade principal do IAM deve ser capaz de listar e descrever clusters para usar o comando AWS CLI da `update-kubeconfig`.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

Usar funções vinculadas ao serviço para o Amazon EKS

O Amazon Elastic Kubernetes Service usa as Funções vinculadas ao serviço do AWS Identity and Access Management IAM. A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Tópicos

- Usar funções para clusters do Amazon EKS
- Usar funções para grupos de nós do Amazon EKS
- Usar funções para os perfis do Amazon EKS Fargate
- Usar funções para conectar um cluster do Kubernetes ao Amazon EKS
- Usar funções para clusters locais do Amazon EKS no Outpost

Usar funções para clusters do Amazon EKS

O Amazon Elastic Kubernetes Service usa as Funções vinculadas ao serviço do AWS Identity and Access Management IAM. A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das

funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros produtos que oferecem suporte às funções vinculadas a serviços, consulte Produtos da AWS compatíveis com o IAM e procure os produtos que apresentam Yes (Sim) na coluna Service-linked role (Função vinculada a serviço). Escolha um Sim com um link para visualizar a documentação do perfil vinculado a esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa a função vinculada ao serviço chamada `AWSServiceRoleForAmazonEKS` – A função permite que o Amazon EKS gerencie clusters em sua conta. As políticas anexadas permitem que a função gerencie os seguintes recursos: interfaces de rede, grupos de segurança, logs e VPCs.

> **Note**
>
> A função vinculada ao serviço `AWSServiceRoleForAmazonEKS` é distinta da função necessária para a criação do cluster. Para ter mais informações, consulte Função do IAM do cluster do Amazon EKS.

A função vinculada ao serviço `AWSServiceRoleForAmazonEKS` confia nos seguintes serviços para assumir a função:

- `eks.amazonaws.com`

A política de permissões da função permite que o Amazon EKS conclua as seguintes ações nos recursos especificados:

- `AmazonEKSServiceRolePolicy`

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada a serviço. Para obter mais informações, consulte Permissões de perfil vinculado a serviços no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon EKS

Não é necessário criar manualmente uma função vinculada a serviço. Quando você cria um cluster no AWS Management Console, na AWS CLI ou na API da AWS, o Amazon EKS cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, você poderá usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria um cluster, o Amazon EKS cria uma função vinculada ao serviço para você novamente.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKS`. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição da função usando o IAM. Para ter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um recurso ou serviço que exija uma função vinculada a um serviço, recomendamos que você exclua essa função. Dessa forma, você não terá uma entidade não utilizada que não seja ativamente monitorada ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Para excluir recursos do Amazon EKS usados pelo**`AWSServiceRoleForAmazonEKS`**Função do .

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, escolha Clusters.

3. Se o cluster tiver grupos de nós ou perfis do Fargate, você deverá excluí-los antes de excluir o cluster. Para obter mais informações, consulte Excluir um grupo de nós gerenciados do seu cluster e Excluir um perfil do Fargate.
4. NoClustersEscolha o cluster que você quer excluir e escolhaExcluir.
5. Digite o nome do cluster na janela de confirmação de exclusão e escolha Delete (Excluir) para excluir.
6. Repita esse procedimento para qualquer outro cluster na sua conta. Aguarde até que todas as operações de exclusão sejam concluídas.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API da AWS para excluir o perfil vinculado ao serviço `AWSServiceRoleForAmazonEKS`. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte a funções vinculadas a serviço do Amazon EKS

O Amazon EKS oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço estiver disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon EKS.

Usar funções para grupos de nós do Amazon EKS

O Amazon EKS usa funções vinculadas ao serviço do AWS Identity and Access Management IAM. A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros produtos que oferecem suporte às funções vinculadas a serviços, consulte Produtos da AWS compatíveis com o IAM e procure os produtos que apresentam

Yes (Sim) na coluna Service-linked role (Função vinculada a serviço). Escolha um Sim com um link para visualizar a documentação do perfil vinculado a esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa a função vinculada ao serviço chamada `AWSServiceRoleForAmazonEKSNodegroup` – A função permite que o Amazon EKS gerencie grupos de nós em sua conta. As políticas anexadas permitem que a função gerencie os seguintes recursos: grupos de Auto Scaling, grupos de segurança, modelos de execução e perfis de instância do IAM.

A função vinculada ao serviço `AWSServiceRoleForAmazonEKSNodegroup` confia nos seguintes serviços para aceitar a função:

- `eks-nodegroup.amazonaws.com`

A política de permissões da função permite que o Amazon EKS conclua as seguintes ações nos recursos especificados:

- `AWSServiceRoleForAmazonEKSNodegroup`

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada a serviço. Para obter mais informações, consulte Permissões de perfil vinculado a serviços no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon EKS

Não é necessário criar manualmente uma função vinculada a serviço. Quando você cria um grupo de nós gerenciados no AWS Management Console, na AWS CLI ou com a API da AWS, o Amazon EKS cria a função vinculada ao serviço para você.

> **Important**
>
> Esse perfil vinculado ao serviço pode aparecer em sua conta se você concluiu uma ação em outro serviço que usa os atributos compatíveis com esse perfil. Se você estava usando o serviço do Amazon EKS antes de 1º de janeiro de 2017, quando ele começou a oferecer suporte a funções vinculadas ao serviço, o Amazon EKS criou a função AWSServiceRoleForAmazonEKSNodegroup na sua conta. Para saber mais, consulte Uma nova função apareceu na minha conta do IAM.

Criação de uma função vinculada ao serviço no Amazon EKS (API da AWS)

Não é necessário criar manualmente uma função vinculada a serviço. Quando você cria um grupo de nós gerenciados no AWS Management Console, na AWS CLI ou com a API da AWS, o Amazon EKS cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, você poderá usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria outro grupo de nós gerenciados, o Amazon EKS cria a função vinculada ao serviço novamente para você.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKSNodegroup`. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição da função usando o IAM. Para ter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um recurso ou serviço que exija uma função vinculada a um serviço, recomendamos que você exclua essa função. Dessa forma, você não terá uma entidade não utilizada que não seja ativamente monitorada ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Para excluir recursos do Amazon EKS usados pelo**`AWSServiceRoleForAmazonEKSNodegroup`**Função do .

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

2. No painel de navegação à esquerda, escolha Clusters.
3. Selecione a guia Compute (Computação).
4. Na seção Node groups (Grupos de nós), escolha o grupo de nós a ser excluído.
5. Digite o nome do grupo de nós na janela de confirmação de exclusão e selecione Delete (Excluir) para excluir.
6. Repita esse procedimento para qualquer outro grupo de nós no cluster. Aguarde até que todas as operações de exclusão sejam concluídas.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API da AWS para excluir o perfil vinculado ao serviço `AWSServiceRoleForAmazonEKSNodegroup`. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte a funções vinculadas a serviço do Amazon EKS

O Amazon EKS oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço estiver disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon EKS.

Usar funções para os perfis do Amazon EKS Fargate

O Amazon EKS usa funções vinculadas ao serviço do AWS Identity and Access Management IAM. A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros produtos que oferecem suporte às funções vinculadas a serviços, consulte Produtos da AWS compatíveis com o IAM e procure os produtos que apresentam

Yes (Sim) na coluna Service-linked role (Função vinculada a serviço). Escolha um Sim com um link para visualizar a documentação do perfil vinculado a esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa a função vinculada ao serviço denominada `AWSServiceRoleForAmazonEKSForFargate`: a função permite que o Amazon EKS Fargate configure as redes da VPC necessárias para Pods do Fargate. As políticas anexadas permitem que a função crie e exclua interfaces de rede elástica e descreva recursos e interfaces de rede elástica.

A função vinculada ao serviço `AWSServiceRoleForAmazonEKSForFargate` confia nos seguintes serviços para aceitar a função:

- `eks-fargate.amazonaws.com`

A política de permissões da função permite que o Amazon EKS conclua as seguintes ações nos recursos especificados:

- `AmazonEKSForFargateServiceRolePolicy`

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada a serviço. Para obter mais informações, consulte Permissões de perfil vinculado a serviços no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon EKS

Não é necessário criar manualmente uma função vinculada a serviço. Quando você cria um perfil do Fargate no AWS Management Console, na AWS CLI ou com a API da AWS, o Amazon EKS cria a função vinculada ao serviço para você.

> **Important**
>
> Esse perfil vinculado ao serviço pode aparecer em sua conta se você concluiu uma ação em outro serviço que usa os atributos compatíveis com esse perfil. Se você estava usando o serviço do Amazon EKS antes de 13 de dezembro de 2019, quando ele começou a oferecer suporte a funções vinculadas ao serviço, o Amazon EKS criou a função AWSServiceRoleForAmazonEKSForFargate na sua conta. Para saber mais, consulte Uma nova função apareceu na minha conta do IAM.

Criação de uma função vinculada ao serviço no Amazon EKS (API da AWS)

Não é necessário criar manualmente uma função vinculada a serviço. Quando você cria um perfil do Fargate no AWS Management Console, na AWS CLI ou com a API da AWS, o Amazon EKS cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, você poderá usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria outro grupo de nós gerenciados, o Amazon EKS cria a função vinculada ao serviço novamente para você.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKSForFargate`. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição da função usando o IAM. Para ter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um recurso ou serviço que exija uma função vinculada a um serviço, recomendamos que você exclua essa função. Dessa forma, você não terá uma entidade não utilizada que não seja ativamente monitorada ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Para excluir recursos do Amazon EKS usados pelo**`AWSServiceRoleForAmazonEKSForFargate`**Função do .

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

2. No painel de navegação à esquerda, escolha Clusters.
3. Na página Clusters, selecione seu cluster.
4. Selecione a guia Compute (Computação).
5. Se houver algum perfil do Fargate na seção Fargate profiles (Perfis do Fargate), selecione cada um individualmente e, depois, Delete (Excluir).
6. Digite o nome do perfil na janela de confirmação de exclusão e escolha Delete (Excluir) para excluir.
7. Repita o procedimento para todos os perfis do Fargate no cluster e para outros clusters na sua conta.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API da AWS para excluir a função vinculada ao serviço AWSServiceRoleForAmazonEKSForFargate. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte a funções vinculadas a serviço do Amazon EKS

O Amazon EKS oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço estiver disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon EKS.

Usar funções para conectar um cluster do Kubernetes ao Amazon EKS

O Amazon EKS usa funções vinculadas ao serviço do AWS Identity and Access Management IAM. A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros produtos que oferecem suporte às funções vinculadas a serviços, consulte Produtos da AWS compatíveis com o IAM e procure os produtos que apresentam Yes (Sim) na coluna Service-linked role (Função vinculada a serviço). Escolha um Sim com um link para visualizar a documentação do perfil vinculado a esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa a função vinculada ao serviço denominada `AWSServiceRoleForAmazonEKSConnector`: a função permite que o Amazon EKS se conecte com os clusters do Kubernetes. As políticas anexadas permitem que a função gerencie os recursos necessários para se conectar ao seu cluster do Kubernetes registrado.

A função vinculada ao serviço `AWSServiceRoleForAmazonEKSConnector` confia nos seguintes serviços para aceitar a função:

- `eks-connector.amazonaws.com`

A política de permissões da função permite que o Amazon EKS conclua as seguintes ações nos recursos especificados:

- `AmazonEKSConnectorServiceRolePolicy`

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada a serviço. Para obter mais informações, consulte Permissões de perfil vinculado a serviços no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon EKS

Você não precisa criar manualmente uma função vinculada a serviço para se conectar a um cluster. Quando você conecta um cluster ao AWS Management Console, à AWS CLI, ao `eksctl` ou à API da AWS, o Amazon EKS cria a função vinculada a serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, você poderá usar esse mesmo processo para recriar o perfil em sua conta. Quando você conecta um cluster, o Amazon EKS cria uma função vinculada a serviço para você novamente.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKSConnector`. Depois que criar um perfil vinculado ao serviço, você

não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição da função usando o IAM. Para ter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um recurso ou serviço que exija uma função vinculada a um serviço, recomendamos que você exclua essa função. Dessa forma, você não terá uma entidade não utilizada que não seja ativamente monitorada ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Para excluir recursos do Amazon EKS usados pelo**`AWSServiceRoleForAmazonEKSConnector`**Função do .

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, escolha Clusters.
3. Na página Clusters, selecione seu cluster.
4. Selecione a guia Deregister (Cancelar registro) e, em seguida, selecione a guia OK.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API da AWS para excluir a função vinculada ao serviço AWSServiceRoleForAmazonEKSForFargate. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Usar funções para clusters locais do Amazon EKS no Outpost

O Amazon Elastic Kubernetes Service usa as Funções vinculadas ao serviço do AWS Identity and Access Management IAM. A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros produtos que oferecem suporte às funções vinculadas a serviços, consulte Produtos da AWS compatíveis com o IAM e procure os produtos que apresentam Yes (Sim) na coluna Service-linked role (Função vinculada a serviço). Escolha um Sim com um link para visualizar a documentação do perfil vinculado a esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa a função vinculada ao serviço denominada `AWSServiceRoleForAmazonEKSLocalOutpost`: a função permite que o Amazon EKS gerencie clusters na sua conta. As políticas anexadas permitem que a função gerencie os seguintes recursos: interfaces de rede, grupos de segurança, logs e instâncias do Amazon EC2.

> **Note**
>
> A função vinculada ao serviço `AWSServiceRoleForAmazonEKSLocalOutpost` é distinta da função necessária para a criação do cluster. Para ter mais informações, consulte Função do IAM do cluster do Amazon EKS.

A função vinculada ao serviço `AWSServiceRoleForAmazonEKSLocalOutpost` confia nos seguintes serviços para assumir a função:

- `outposts.eks-local.amazonaws.com`

A política de permissões da função permite que o Amazon EKS conclua as seguintes ações nos recursos especificados:

- `AmazonEKSServiceRolePolicy`

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada a serviço. Para obter mais informações, consulte Permissões de perfil vinculado a serviços no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon EKS

Não é necessário criar manualmente uma função vinculada a serviço. Quando você cria um cluster no AWS Management Console, na AWS CLI ou na API da AWS, o Amazon EKS cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, você poderá usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria um cluster, o Amazon EKS cria uma função vinculada ao serviço para você novamente.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKSLocalOutpost`. Depois que você criar um perfil vinculado ao serviço, não poderá alterar o nome do perfil, pois várias entidades podem fazer referência ao perfil. No entanto, você poderá editar a descrição do perfil usando o IAM. Para ter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um recurso ou serviço que exija uma função vinculada a um serviço, recomendamos que você exclua essa função. Dessa forma, você não terá uma entidade não utilizada que não seja ativamente monitorada ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Para excluir recursos do Amazon EKS usados pelo**`AWSServiceRoleForAmazonEKSLocalOutpost`**Função do .

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, escolha Clusters do Amazon EKS.
3. Se o cluster tiver grupos de nós ou perfis do Fargate, você deverá excluí-los antes de excluir o cluster. Para obter mais informações, consulte Excluir um grupo de nós gerenciados do seu cluster e Excluir um perfil do Fargate.
4. NoClustersEscolha o cluster que você quer excluir e escolhaExcluir.
5. Digite o nome do cluster na janela de confirmação de exclusão e escolha Delete (Excluir) para excluir.
6. Repita esse procedimento para qualquer outro cluster na sua conta. Aguarde até que todas as operações de exclusão sejam concluídas.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API da AWS para excluir o perfil vinculado ao serviço `AWSServiceRoleForAmazonEKSLocalOutpost`. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte a funções vinculadas a serviço do Amazon EKS

O Amazon EKS oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço estiver disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon EKS.

Função do IAM do cluster do Amazon EKS

A função do IAM de cluster do Amazon EKS é necessária para cada cluster. slusters do Kubernetes gerenciados pelo Amazon EKS usam essa função para gerenciar nós e o provedor de nuvem legado usa essa função para criar balanceadores de carga com o Elastic Load Balancing para serviços.

Antes de criar clusters do Amazon EKS, você deve criar uma função do IAM com uma das seguintes políticas do IAM:

- AmazonEKSClusterPolicy
- Uma política do IAM personalizada. As permissões mínimas a seguir permitem que o Kubernetes cluster gerencie nós, mas não permitem que o provedor de nuvem legado crie balanceadores de carga com o Elastic Load Balancing. Sua política do IAM personalizada deve ter pelo menos as seguintes permissões:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": "kubernetes.io/cluster/*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeAvailabilityZones",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Antes de 3 de outubro de 2023, AmazoneksClusterPolicy era exigido no perfil do IAM para cada cluster.
> Antes de 16 de abril de 2020, AmazonEKSServicePolicy e AmazonEKSClusterPolicy eram obrigatórios, e o nome sugerido para o perfil era `eksServiceRole`. Com o perfil vinculado ao serviço `AWSServiceRoleForAmazonEKS`, a política AmazonEKSServicePolicy não será mais necessária para clusters criados em ou após 16 de abril de 2020.

Verificar se há uma função de cluster existente

É possível usar o procedimento a seguir para verificar se a conta já tem a função do cluster do Amazon EKS.

Para verificar a **`eksClusterRole`** no console do IAM;

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.
3. Pesquise `eksClusterRole` na lista de funções. Se não houver uma função que inclua `eksClusterRole`, consulte Criar uma função para o cluster do Amazon EKS para criar a função. Se houver uma função que inclua `eksClusterRole`, selecione-a para visualizar as políticas anexadas.
4. Escolha Permissões.
5. Verifique se a política gerenciada AmazonEKSClusterPolicy está anexada à função. Se a política estiver anexada, a função do cluster do Amazon EKS estará configurada corretamente.
6. Escolha Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).
7. Verifique se o relacionamento de confiança contém a seguinte política: Se o relacionamento de confiança corresponder à seguinte política, escolha Cancel (Cancelar). Se o relacionamento de confiança não corresponder, copie a política para a janela Edit trust policy (Editar política de confiança) e escolha Update policy (Atualizar política).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Criar uma função para o cluster do Amazon EKS

Para criar uma função do cluster, você pode usar o AWS Management Console ou o AWS CLI.

AWS Management Console

Para criar a função de cluster do Amazon EKS no console do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Escolha Roles (Funções) e, em seguida, Create Role (Criar função).
3. Em Tipo de entidade confiável, selecione Serviço da AWS.
4. Na lista suspensa Casos de uso para outros Serviços da AWS, escolha EKS.
5. Escolha EKS - Cluster para o caso de uso e escolha Next (Próximo).
6. Na guia Add permissions (Adicionar permissões), escolha Next (Próximo).
7. Em Role name (Nome da função), insira um nome exclusivo para a função, como **`eksClusterRole`**.
8. Em Description (Descrição), insira um texto descritivo, como **`Amazon EKS - Cluster role`**.
9. Selecione Criar função.

AWS CLI

1. Copie o conteúdo a seguir em um arquivo denominado *`cluster-trust-policy.json`*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Crie a função. É possível substituir ***eksClusterRole*** por qualquer nome que você escolher.

```
aws iam create-role \
  --role-name eksClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. Anexe a política do IAM necessária à função.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name eksClusterRole
```

Perfil do IAM em nós do Amazon EKS

O daemon `kubelet` do nó do Amazon EKS chama as APIs da AWS em seu nome. Os nós recebem permissões para essas chamadas de API por meio de um perfil de instância do IAM e políticas associadas. Antes de iniciar os nós e registrá-los em um cluster, você deve criar um perfil do IAM para uso desses nós quando eles forem iniciados. Esse requisito se aplica a nós executados com a AMI otimizada para Amazon EKS fornecida pela Amazon ou com qualquer outra AMI do nó que você pretende usar. Além disso, este requisito se aplica tanto aos grupos de nós gerenciados quanto aos nós autogerenciados.

> **Note**
>
> Não é possível usar a mesma função usada para criar clusters.

Antes de criar os nós, é necessário criar um perfil do IAM com as seguintes permissões:

- Permissões para que o `kubelet` descreva os recursos do Amazon EC2 na VPC, conforme fornecido pela política `AmazonEKSWorkerNodePolicy`. Esta política também fornece as permissões para o Amazon EKS Pod Identity Agent.

- Permissões para que o `kubelet` use imagens de contêiner do Amazon Elastic Container Registry (Amazon ECR), como as fornecidas pela política `AmazonEC2ContainerRegistryReadOnly`. As permissões para usar imagens de contêiner do Amazon Elastic Container Registry (Amazon ECR) são necessárias porque os complementos integrados para a rede executam pods que usam imagens de contêiner do Amazon ECR.
- (Opcional) Permissões para que o Amazon EKS Pod Identity Agent use a ação `eks-auth:AssumeRoleForPodIdentity` para recuperar credenciais para pods. Se você não usa o AmazonEKSWorkerNodePolicy, forneça esta permissão além das permissões do EC2 para usar o EKS Pod Identity.
- (Opcional) Se você não usa o IRSA ou o EKS Pod Identity para conceder permissões aos pods da VPC CNI, forneça permissões para a VPC CNI na função da instância. É possível usar tanto a política gerenciada `AmazonEKS_CNI_Policy` (caso você tenha criado o seu cluster com a família `IPv4`) ou uma política IPv6 criada por você (caso tenha criado o seu cluster com a família `IPv6`). Contudo, em vez de anexar a política à essa função, recomendamos que você anexe a política a uma função separada usada especificamente para o complemento CNI da Amazon VPC. Para obter mais informações sobre a criação de uma função separada para o complemento CNI da Amazon VPC, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).

> **Note**
>
> Antes de 3 de outubro de 2023, `AmazonEKSWorkerNodePolicy` e `AmazonEC2ContainerRegistryReadOnly` eram obrigatórios no perfil do IAM para cada grupo de nós gerenciados.
> Os grupos de nós do Amazon EC2 devem ter uma função do IAM diferente do perfil do Fargate. Para ter mais informações, consulte Perfil do IAM para execução de Pod do Amazon EKS.

Verificar se há uma função existente do nó

Use o procedimento a seguir para verificar se a conta já tem a função de nó do Amazon EKS.

Para verificar a **`eksNodeRole`** no console do IAM;

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.

3. Procure `eksNodeRole`, `AmazonEKSNodeRole` ou `NodeInstanceRole` na lista de funções. Se não houver uma função com um desses nomes, consulte Criação de uma função para a função do IAM de nó do Amazon EKS para criar a função. Se houver uma função que contenha `eksNodeRole`, `AmazonEKSNodeRole` ou `NodeInstanceRole`, selecione-a para visualizar as políticas anexadas.
4. Escolha Permissões.
5. Assegure-se de que as políticas gerenciadas AmazonEKSWorkerNodePolicy e AmazonEC2ContainerRegistryReadOnly estejam anexadas à função ou que uma política personalizada seja anexada com as permissões mínimas necessárias

> **Note**
>
> Se a AmazonEKS_CNI_Policy estiver anexada à função, recomendamos removê-la e anexá-la a um perfil do IAM mapeado para a conta do serviço do Kubernetes do `aws-node`. Para ter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).

6. Escolha Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).
7. Verifique se o relacionamento de confiança contém a seguinte política: Se o relacionamento de confiança corresponder à seguinte política, escolha Cancel (Cancelar). Se o relacionamento de confiança não corresponder, copie a política para a janela Edit trust policy (Editar política de confiança) e escolha Update policy (Atualizar política).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Criação de uma função para a função do IAM de nó do Amazon EKS

É possível criar a função do IAM de nós com o AWS Management Console ou a AWS CLI.

AWS Management Console

Para criar a função do IAM de nó no console do Amazon EKS

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.
3. Na página Roles (Funções), selecione Create role (Criar função).
4. Na página Select trusted entity (Selecionar entidade confiável), faça o seguinte:
   a. Na seção Tipo de entidade confiável, escolha Serviço da AWS.
   b. Em Use case (Caso de uso), selecione EC2.
   c. Escolha Próximo.
5. Na página Add permissions (Adicionar permissões), faça o seguinte:
   a. Na caixa Filter policies (Políticas de filtro) insira **`AmazonEKSWorkerNodePolicy`**.
   b. Marque a caixa de seleção à esquerda de AmazonEKSWorkerNodePolicy nos resultados da pesquisa.
   c. Escolha Clear filters (Limpar filtros).
   d. Na caixa Filter policies (Políticas de filtro) insira **`AmazonEC2ContainerRegistryReadOnly`**.
   e. Marque a caixa de seleção à esquerda de AmazonEC2ContainerRegistryReadOnly nos resultados da pesquisa.

      A política gerenciada AmazonEKS_CNI_Policy ou uma política de IPv6 que você criar também deverá ser anexada a essa função ou a uma função diferente que seja mapeada para a conta de serviço do Kubernetes do `aws-node`. Recomendamos atribuir a política à função associada à conta de serviço do Kubernetes em vez de atribuí-la a essa função. Para ter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).
   f. Escolha Próximo.
6. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:

a. Em Role name (Nome da função), insira um nome exclusivo para a função, como **`AmazonEKSNodeRole`**.

b. Em Description (Descrição), substitua o texto atual por um texto descritivo como **`Amazon EKS - Node role`**.

c. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.

d. Selecione Criar função.

AWS CLI

1. Execute o seguinte comando para criar o arquivo `node-role-trust-relationship.json`.

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

2. Crie o perfil do IAM.

```
aws iam create-role \
  --role-name AmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-relationship.json"
```

3. Anexe duas políticas do IAM gerenciadas necessárias à função do IAM.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
```

```
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name AmazonEKSNodeRole
```

4. Anexe uma das seguintes políticas do IAM à função do IAM, dependendo de com qual família de IP você criou o seu cluster. A política deve ser anexada a essa função ou a uma função associada à conta de serviço do `aws-node` do Kubernetes que é usada para o Amazon VPC CNI plugin for Kubernetes. Recomendamos atribuir a política à função associada à conta de serviço do Kubernetes. Para atribuir a política à função associada à conta de serviço do Kubernetes, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).

   - IPv4

     ```
     aws iam attach-role-policy \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
       --role-name AmazonEKSNodeRole
     ```

   - IPv6

     1. Copie o texto a seguir e salve-o em um arquivo chamado *vpc-cni-ipv6-policy*.json.

        ```
        {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "ec2:AssignIpv6Addresses",
                        "ec2:DescribeInstances",
                        "ec2:DescribeTags",
                        "ec2:DescribeNetworkInterfaces",
                        "ec2:DescribeInstanceTypes"
                    ],
                    "Resource": "*"
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "ec2:CreateTags"
                    ],
        ```

```
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. Crie a política do IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

3. Anexe a política do IAM à função do IAM. Substitua *111122223333* pelo ID da sua conta.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

Perfil do IAM para execução de Pod do Amazon EKS

A função de execução de Pod do Amazon EKS é necessária para executar Pods na infraestrutura do AWS Fargate.

Quando o cluster cria Pods na infraestrutura do AWS Fargate, os componentes que estão em execução na infraestrutura do Fargate devem fazer as chamadas para as APIs da AWS em seu nome. Isso ocorre para que eles possam realizar ações como extrair imagens de contêiner do Amazon ECR ou encaminhar logs para outros serviços da AWS. A função de execução de Pod do Amazon EKS fornece as permissões do IAM para isso.

Ao criar um perfil do Fargate, é necessário especificar uma função de execução de Pod para os componentes do Amazon EKS que são executados na infraestrutura do Fargate usando o perfil. Essa função é adicionada ao Controle de acesso com base em função (RBAC) do Kubernetes do cluster para autorização. Isso permite que o `kubelet` que está sendo executado na infraestrutura do Fargate seja registrado no cluster do Amazon EKS para aparecer no cluster como um nó.

> **Note**
>
> O perfil do Fargate deve ter uma função do IAM diferente dos grupos de nós do Amazon EC2.

> **Important**
>
> Os contêineres em execução no Pod do Fargate não podem assumir as permissões do IAM associadas a uma função de execução de Pod. Para dar aos contêineres de Pod do Fargate permissões de acesso a outros serviços da AWS, você deve usar Perfis do IAM para contas de serviço.

Antes de criar um perfil do Fargate, é necessário criar um perfil do IAM com a `AmazonEKSFargatePodExecutionRolePolicy`.

Verifique se existe uma função de execução de Pod configurada corretamente

É possível usar o procedimento a seguir para verificar se a conta já tem uma função de execução de Pod do Amazon EKS configurada corretamente. Para evitar um problema de segurança de representante confuso, é importante que a função restrinja o acesso com base no `SourceArn`. É possível modificar a função de execução conforme necessário de forma a incluir suporte para perfis do Fargate em outros clusters.

Para verificar se há uma função de execução de Pod do Amazon EKS no console do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.
3. Na página Roles (Funções), procure AmazonEKSFargatePodExecutionRole na lista de funções. Se a função não existir, consulte Criar a função de execução de Pod do Amazon EKS para criá-la. Se a função existir, escolha-a.
4. Na página AmazonEKSFargatePodExecutionRole, faça o seguinte:
   a. Escolha Permissões.
   b. Certifique-se de que a política gerenciada pela Amazon AmazonEKSFargatePodExecutionRolePolicy esteja associada ao perfil.

c. Escolha Relações de Confiança.

d. Escolha Edit trust policy (Editar política de confiança).

5. Na página Edit trust policy (Editar política de confiança), verifique se a relação de confiança contém a política a seguir e tem uma linha para perfis do Fargate no cluster. Em caso afirmativo, escolha Cancel (Cancelar).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Se a política corresponder, mas não tiver uma linha especificando os perfis do Fargate no seu cluster, será possível adicionar a seguinte linha sobre o objeto `ArnLike`. Substitua *region-code* pela Região da AWS na qual seu cluster se encontra, *111122223333* pelo ID da sua conta e *my-cluster* pelo nome do cluster.

```
"aws:SourceArn": "arn:aws:eks:region-code:111122223333:fargateprofile/my-cluster/
*",
```

Se a política não corresponder, copie a política anterior completa para o formulário e escolha Update policy (Atualizar política). Substitua *region-code* pela Região da AWS em que seu cluster está localizado. Se você quiser usar a mesma função em todas as Regiões da AWS da sua conta, substitua *region-code* por *. Substitua *111122223333* pelo ID da conta e *my-cluster* pelo nome do cluster. Se quiser usar a mesma função para todos os clusters da sua conta, substitua *my-cluster* por *.

Criar a função de execução de Pod do Amazon EKS

Se você ainda não tiver a função de execução de Pod do Amazon EKS para o cluster, poderá utilizar o AWS Management Console ou a AWS CLI para criá-la.

AWS Management Console

Para criar um perfil de execução de Pod do AWS Fargate com o AWS Management Console

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.
3. Na página Roles (Funções), selecione Create role (Criar função).
4. Na página Select trusted entity (Selecionar entidade confiável), faça o seguinte:
   a. Na seção Tipo de entidade confiável, escolha Serviço da AWS.
   b. Na lista suspensa Casos de uso para outros Serviços da AWS, escolha EKS.
   c. Escolha EKS - Fargate Pod.
   d. Escolha Próximo.
5. Na página Add permissions (Adicionar permissões), escolha Next (Próximo).
6. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:
   a. Em Role name (Nome da função), insira um nome exclusivo para a função, como **`AmazonEKSFargatePodExecutionRole`**.
   b. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.
   c. Selecione Criar função.
7. Na página Roles (Funções), procure AmazonEKSFargatePodExecutionRole na lista de funções. Selecione o perfil de .
8. Na página AmazonEKSFargatePodExecutionRole, faça o seguinte:
   a. Escolha Relações de Confiança.
   b. Escolha Edit trust policy (Editar política de confiança).
9. Na página Edit trust policy (Editar política de confiança), faça o seguinte:

a. Copie e cole o conteúdo a seguir no formulário Edit trust policy (Editar política de confiança). Substitua *region-code* pela Região da AWS em que o seu cluster se encontra. Se você quiser usar a mesma função em todas as Regiões da AWS da sua conta, substitua *region-code* por *. Substitua *111122223333* pelo ID da conta e *my-cluster* pelo nome do cluster. Se quiser usar a mesma função para todos os clusters da sua conta, substitua *my-cluster* por *.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Escolha Atualizar política.

AWS CLI

Para criar um perfil de execução de Pod do AWS Fargate com o AWS CLI

1. Copie e cole o conteúdo a seguir em um arquivo denominado *pod-execution-role-trust-policy*.json. Substitua *region-code* pela Região da AWS em que o seu cluster se encontra. Se você quiser usar a mesma função em todas as Regiões da AWS da sua conta, substitua *region-code* por *. Substitua *111122223333* pelo ID da conta e *my-cluster* pelo nome do cluster. Se quiser usar a mesma função para todos os clusters da sua conta, substitua *my-cluster* por *.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Crie um perfil do IAM de execução de Pod.

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

3. Anexe a política de IAM gerenciada pelo Amazon EKS à função.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Função do IAM do conector do Amazon EKS

Você pode conectar clusters do Kubernetes para visualizá-los no AWS Management Console. Para se conectar a um cluster do Kubernetes, crie um perfil do IAM.

Verificar se já existe um perfil de conector do EKS

Use o procedimento a seguir para verificar se a conta já tem a função de conector do Amazon EKS.

Para verificar a **`AmazonEKSConnectorAgentRole`** no campo de console do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.

2. No painel de navegação à esquerda, escolha Roles.
3. Pesquise `AmazonEKSConnectorAgentRole` na lista de funções. Se não houver uma função que inclua `AmazonEKSConnectorAgentRole`, consulte Criar a função do IAM para o agente de conector do Amazon EKS para criar a função. Se houver uma função que inclua `AmazonEKSConnectorAgentRole`, selecione-a para visualizar as políticas anexadas.
4. Escolha Permissões.
5. Verifique se a política gerenciada AmazonEKSConnectorAgentPolicy está anexada ao perfil. Se a política estiver anexada, o perfil de conector do Amazon EKS estará configurado corretamente.
6. Escolha Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).
7. Verifique se o relacionamento de confiança contém a seguinte política: Se o relacionamento de confiança corresponder à seguinte política, escolha Cancel (Cancelar). Se o relacionamento de confiança não corresponder, copie a política para a janela Edit trust policy (Editar política de confiança) e escolha Update policy (Atualizar política).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Criar a função do IAM para o agente de conector do Amazon EKS

Você pode usar o AWS Management Console ou o AWS CloudFormation para criar uma função de serviço de agente.

AWS CLI

1. Crie um arquivo denominado `eks-connector-agent-trust-policy.json` que contenha o seguinte JSON para ser usado para a função do IAM:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. Crie um arquivo denominado `eks-connector-agent-policy.json` que contenha o seguinte JSON para ser usado para a função do IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SsmControlChannel",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel"
            ],
            "Resource": "arn:aws:eks:*:*:cluster/*"
        },
        {
            "Sid": "ssmDataplaneOperations",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenDataChannel",
                "ssmmessages:OpenControlChannel"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

3. Crie a função de agente do Amazon EKS Connector usando a política de confiança e a política que você criou nos itens de lista anteriores.

```
aws iam create-role \
     --role-name AmazonEKSConnectorAgentRole \
     --assume-role-policy-document file://eks-connector-agent-trust-policy.json
```

4. Anexe a política à função do agente do Amazon EKS Connector.

```
aws iam put-role-policy \
     --role-name AmazonEKSConnectorAgentRole \
     --policy-name AmazonEKSConnectorAgentPolicy \
     --policy-document file://eks-connector-agent-policy.json
```

AWS CloudFormation

Para criar a função do IAM de agente do conector do Amazon EKS com o AWS CloudFormation.

1. Salve o modelo do AWS CloudFormation a seguir em um arquivo de texto no sistema local.

> **Note**
>
> Esse modelo também cria a função vinculada ao serviço que seria criada quando a API `registerCluster` fosse chamada. Para mais detalhes, consulte Usar funções para conectar um cluster do Kubernetes ao Amazon EKS.

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Provisions necessary resources needed to register clusters in EKS'
Parameters: {}
Resources:
  EKSConnectorSLR:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: eks-connector.amazonaws.com
```

```
  EKSConnectorAgentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action: [ 'sts:AssumeRole' ]
            Principal:
              Service: 'ssm.amazonaws.com'

  EKSConnectorAgentPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: EKSConnectorAgentPolicy
      Roles:
        - {Ref: 'EKSConnectorAgentRole'}
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateControlChannel' ]
            Resource:
            - Fn::Sub: 'arn:${AWS::Partition}:eks:*:*:cluster/*'
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateDataChannel',
 'ssmmessages:OpenDataChannel', 'ssmmessages:OpenControlChannel' ]
            Resource: "*"
Outputs:
  EKSConnectorAgentRoleArn:
    Description: The agent role that EKS connector uses to communicate with
 Serviços da AWS.
    Value: !GetAtt EKSConnectorAgentRole.Arn
```

2. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.
3. Escolha Create stack (Criar pilha), com novos recursos ou recursos existentes.
4. Para Specify template (Especificar modelo), selecione Upload a template file (Fazer upload de um arquivo de modelo) e depois Choose file (Escolher arquivo).
5. Selecione o arquivo que você criou anteriormente e, em seguida, selecione Next (Próximo).
6. Em Stack name (Nome da pilha), insira um nome para a função, como `eksConnectorAgentRole`, e selecione Next (Próximo).

7. Na página Configurar opções de pilha, selecione Avançar.
8. Na página Review (Revisão), reveja as informações, confirme se a pilha pode criar recursos do IAM e, em seguida, selecione Create (Criar).

Políticas gerenciadas da AWS para o Amazon Elastic Kubernetes Service

Uma política gerenciada pela AWS é uma política independente criada e administrada pela AWS. As políticas gerenciadas pela AWS são criadas para fornecer permissões a vários casos de uso comuns a fim de que você possa começar a atribuir permissões a usuários, grupos e perfis.

Lembre-se de que as políticas gerenciadas pela AWS podem não conceder permissões de privilégio mínimo para seus casos de uso específicos, por estarem disponíveis para uso por todos os clientes da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da específicas para seus casos de uso.

Você não pode alterar as permissões definidas em políticas gerenciadas AWS. Se AWS atualiza as permissões definidas em um política gerenciada por AWS, a atualização afeta todas as identidades de entidades principais (usuários, grupos e perfis) às quais a política estiver vinculada. É provável que AWS atualize uma política gerenciada por AWS quando um novo Serviço da AWS for lançado, ou novas operações de API forem disponibilizadas para os serviços existentes.

Para obter mais informações, consulte Políticas gerenciadas pela AWS no Guia do usuário do IAM.

Política gerenciada da AWS: AmazonEKS_CNI_Policy

Você pode anexar a `AmazonEKS_CNI_Policy` às suas entidades do IAM. Antes de criar um grupo de nós do Amazon EC2, essa política deve ser anexada à função do IAM do nó, ou para uma função do IAM que seja usada especificamente pelo Amazon VPC CNI plugin for Kubernetes. Isso é para que ele possa realizar ações em seu nome. Recomendamos que você anexe a política a uma função que é usada apenas pelo plugin. Para ter mais informações, consulte Trabalhando com o complemento Amazon VPC CNI plugin for Kubernetes do Amazon EKS e Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).

Detalhes da permissão

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2:*NetworkInterface`** e **`ec2:*PrivateIpAddresses`**: permite que o plug-in CNI da Amazon VPC execute ações, como o provisionamento de interfaces de rede elástica e de endereços IP para Pods, com a finalidade de fornecer um sistema de rede para aplicações executadas no Amazon EKS.
- Ações de leitura do **`ec2`**: permite que o plug-in CNI da Amazon VPC execute ações, como a descrição de instâncias e de sub-redes, para visualizar a quantidade de endereços IP disponíveis nas sub-redes da Amazon VPC. O plug-in CNI da VPC pode usar os endereços IP disponíveis em cada sub-rede para escolher as sub-redes com a maior quantidade de endereços IP disponíveis para usar ao criar uma interface de rede elástica.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKS_CNI_Policy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AmazonEKSClusterPolicy

Você pode anexar `AmazonEKSClusterPolicy` às entidades do IAM. Antes de criar um cluster, é necessário ter um perfil do IAM de cluster com essa política anexada. Os clusters do Kubernetes gerenciados pelo Amazon EKS fazem chamadas para outros serviços da AWS em seu nome. Eles fazem isso para gerenciar os recursos que você usa com o serviço do.

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`autoscaling`** – Leia e atualize a configuração de um grupo do Auto Scaling. Essas permissões não são usadas pelo Amazon EKS, mas permanecem na política para compatibilidade com versões anteriores.
- **`ec2`** – Trabalhe com volumes e recursos de rede associados aos nós do Amazon EC2. Isso é necessário para que o ambiente de gerenciamento do Kubernetes possa unir instâncias a um cluster e provisionar e gerenciar dinamicamente os volumes do Amazon EBS solicitados por volumes persistentes do Kubernetes.
- **`elasticloadbalancing`** – Trabalhe com o Elastic Load Balancers e adicione nós a eles como destinos. Isso é necessário para que o ambiente de gerenciamento do Kubernetes possa provisionar dinamicamente os Elastic Load Balancers solicitados por serviços do Kubernetes.
- **`iam`** – Crie uma função vinculada a serviço. Isso é necessário para que o ambiente de gerenciamento do Kubernetes possa provisionar dinamicamente os Elastic Load Balancers que são solicitados por serviços do Kubernetes.

- **`kms`** – Leia uma chave de AWS KMS. Isso é necessário para que o ambiente de gerenciamento do Kubernetes seja compatível com a criptografia de segredo dos segredos do Kubernetes armazenados em `etcd`.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSClusterPolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AmazonEKSFargatePodExecutionRolePolicy

Você pode anexar `AmazonEKSFargatePodExecutionRolePolicy` às entidades do IAM. Antes de criar um perfil do Fargate, você deverá criar uma função de execução de Pod do Fargate e anexar essa política a ela. Para ter mais informações, consulte Etapa 2: criar um perfil de execução de Pod do Fargate e Defina quais Pods usarão o AWS Fargate quando em execução.

Essa política concede à função as permissões que fornecem acesso a outros recursos de serviço do AWS necessários para executar Pods do Amazon EKS no Fargate.

Detalhes da permissão

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ecr`**: permite que pods em execução no Fargate extraiam imagens de contêiner armazenadas no Amazon ECR.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSFargatePodExecutionRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AmazonEKSForFargateServiceRolePolicy

Não é possível anexar `AmazonEKSForFargateServiceRolePolicy` às suas entidades do IAM. Esta política é anexada a uma função vinculada ao serviço que permite ao Amazon EKS realizar ações em seu nome. Para obter mais informações, consulte AWSServiceRoleforAmazonEKSForFargate.

Esta política concede permissões necessárias ao Amazon EKS para executar tarefas do Fargate. A política só é usada se você tiver nós Fargate.

Detalhes da permissão

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`** –Crie e exclua interfaces de rede elástica e descreva os recursos e interfaces de rede elástica. Isso é necessário para que o serviço Amazon EKS Fargate possa configurar a rede VPC necessária para Fargate Pods.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSForFargateServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AmazonEKSServicePolicy

Você pode anexar `AmazonEKSServicePolicy` às entidades do IAM. Os clusters criados antes de 16 de abril de 2020 exigiam que você criasse uma função do IAM e associasse essa política à ela. Os clusters criados em ou após 16 de abril de 2020 não exigem que você crie uma função, nem que você atribua essa política. Quando você cria um cluster usando uma entidade principal do IAM que tem a permissão da `iam:CreateServiceLinkedRole`, a função vinculada ao serviço do AWSServiceRoleforAmazonEKS é recriada automaticamente para você. A função vinculada ao serviço tem oPolítica gerenciada da AWS: AmazonEKSServiceRolePolicyAnexado a ele.

Essa política permite que o Amazon EKS crie e gerencie os recursos necessários para operar clusters do Amazon EKS.

Detalhes da permissão

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`eks`**: atualizar a versão do Kubernetes do cluster depois que você iniciar uma atualização. Essa permissão não é usada pelo Amazon EKS, mas permanece na política para compatibilidade com versões anteriores.
- **`ec2`** – Trabalhe com interfaces de rede elástica e outros recursos e etiquetas de rede. Isso é exigido pelo Amazon EKS para configurar redes que facilitem a comunicação entre nós e o ambiente de gerenciamento do Kubernetes.
- **`route53`** – Associe uma VPC a uma zona hospedada. Isso é exigido pelo Amazon EKS para habilitar a rede privada de endpoint para o servidor de API de cluster do Kubernetes.
- **`logs`** - Eventos de log Isso é necessário para que o Amazon EKS possa enviar logs do ambiente de gerenciamento do Kubernetes para o CloudWatch.
- **`iam`** – Crie uma função vinculada a serviço. Isso é necessário para que o Amazon EKS crie a função vinculada ao serviço `AWSServiceRoleForAmazonEKS` em seu nome.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSServicePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AmazonEKSServiceRolePolicy

Não é possível anexar `AmazonEKSServiceRolePolicy` às suas entidades do IAM. Esta política é anexada a uma função vinculada ao serviço que permite ao Amazon EKS realizar ações em seu nome. Para ter mais informações, consulte Permissões de função vinculada ao serviço para o Amazon EKS. Quando você cria um cluster usando uma entidade principal do IAM que tem a permissão da `iam:CreateServiceLinkedRole`, a função vinculada ao serviço do AWSServiceRoleforAmazonEKS é recriada automaticamente para você e essa política é anexada a ela.

Esta política permite que a função vinculada ao serviço chame serviços da AWS da em seu nome.

Detalhes da permissão

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`**: crie e descreva as interfaces de rede elástica e as instâncias do Amazon EC2, o grupo de segurança de cluster e VPC que são necessários para a criação de um cluster.
- **`iam`** – Lista todas as políticas gerenciadas anexadas a uma função do IAM. Isso é necessário para que o Amazon EKS possa listar e validar todas as políticas gerenciadas e permissões necessárias para criar um cluster.
- Associar uma VPC a uma zona hospedada: isso é necessário para que o Amazon EKS habilite a rede privada de endpoint para o servidor de API de cluster do Kubernetes.
- Registrar evento em log: isso é necessário para que o Amazon EKS possa enviar logs do ambiente de gerenciamento do Kubernetes para o CloudWatch.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AmazonEKSVPCResourceController

É possível anexar a política `AmazonEKSVPCResourceController` a suas identidades do IAM. Se você estiver usando grupos de segurança para Pods, anexe essa política ao Função do IAM do cluster do Amazon EKS para que ele realize ações em seu nome.

Esta política concede à função de cluster permissões para gerenciar interfaces de rede elástica e endereços IP para nós.

Detalhes da permissão

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`**: gerenciar interfaces de rede elástica e endereços IP para compatibilidade com grupos de segurança de Pod e nós do Windows.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSVPCResourceController no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AmazonEKSWorkerNodePolicy

Você pode anexar a `AmazonEKSWorkerNodePolicy` às suas entidades do IAM. É preciso anexar essa política a um perfil do IAM do nó que você especifica ao criar nós do Amazon EC2 que permitem ao Amazon EKS realizar ações em seu nome. Se você criar um grupo de nós usando`eksctl`, ele cria a função do nó do IAM e anexa essa política à função automaticamente.

Esta política concede aos nós Amazon EKS Amazon EC2 permissões para se conectar a clusters do Amazon EKS.

Detalhes da permissão

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`** – Leia o volume da instância e as informações de rede. Isso é necessário para que os nós do Kubernetes possam descrever informações sobre os recursos do Amazon EC2 necessários para que o nó faça parte do cluster do Amazon EKS.
- **`eks`**: opcionalmente, descreva o cluster como parte do bootstrapping do nó.
- **`eks-auth:AssumeRoleForPodIdentity`**: permita a recuperação de credenciais para workloads do EKS no nó. Essa API é necessária para o EKS Pod Identity funcionar corretamente.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSWorkerNodePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AWSServiceRoleForAmazonEKSNodegroup

Não é possível anexar `AWSServiceRoleForAmazonEKSNodegroup` às suas entidades do IAM. Esta política é anexada a uma função vinculada ao serviço que permite ao Amazon EKS realizar ações em seu nome. Para ter mais informações, consulte Permissões de função vinculada ao serviço para o Amazon EKS.

Esta política concede o`AWSServiceRoleForAmazonEKSNodegroup`permissões de função que permitem que ele crie e gerencie grupos de nós do Amazon EC2 em sua conta.

Detalhes da permissão

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`**: trabalhe com grupos de segurança, tags, reservas de capacidade e modelos de execução. Isso é necessário para grupos de nós gerenciados pelo Amazon EKS a fim de habilitar a configuração de acesso remoto e descrever reservas de capacidade que podem ser usadas em grupos de nós gerenciados. Além disso, os grupos de nós gerenciados do Amazon EKS criam um modelo de execução em seu nome. Isso serve para configurar o grupo do Amazon EC2 Auto Scaling que oferece suporte a cada grupo de nós gerenciados.
- **`iam`** – Crie uma função vinculada a serviço e transmita uma função. Isso é exigido pelos grupos de nós gerenciados do Amazon EKS para gerenciar perfis de instância para a função que está sendo passada ao criar um grupo de nós gerenciados. Esse perfil de instância é usado pelas instâncias do Amazon EC2 executadas como parte de um grupo de nós gerenciados. O Amazon EKS precisa criar funções vinculadas ao serviço para outros serviços, como os grupos do Amazon EC2 Auto Scaling. Essas permissões são usadas na criação de um grupo de nós gerenciados
- **`autoscaling`** – Trabalhe com grupos de Auto Scaling de segurança. Isso é exigido pelos grupos de nós gerenciados do Amazon EKS para gerenciar o grupo do Amazon EC2 Auto Scaling que apoia cada grupo de nós gerenciados. Ele também é usado para compatibilidade com as funcionalidades, como remover Pods quando os nós são encerrados ou reciclados durante atualizações de grupo de nós.

Para visualizar a versão mais recente do documento de política JSON, consulte AWSServiceRoleForAmazonEKSNodegroup no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AmazonEBSCSIDriverPolicy

A política `AmazonEBSCSIDriverPolicy` permite que o driver da Container Storage Interface (CSI) do Amazon EBS crie, modifique, anexe, desconecte e exclua volumes em seu nome. Ela também concede permissões ao driver EBS CSI para criar e excluir snapshots e listar suas instâncias, volumes e snapshots.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEBSCSIDriverServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada da AWS: AmazonEFSCSIDriverPolicy

A política `AmazonEFSCSIDriverPolicy` permite que a Container Storage Interface (CSI) do Amazon EFS crie e exclua pontos de acesso em seu nome. Ela também concede ao driver CSI do Amazon EFS permissões para listar os sistemas de arquivos, destinos de montagem e zonas de disponibilidade do Amazon EC2 dos seus pontos de acesso.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEFSCSIDriverServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada pela AWS: AmazonEKSLocalOutpostClusterPolicy

Essa política pode ser anexada a entidades do IAM. Antes de criar um cluster local, é necessário anexar essa política à função do cluster. Os clusters do Kubernetes gerenciados pelo Amazon EKS fazem chamadas para outros serviços da AWS em seu nome. Eles fazem isso para gerenciar os recursos que você usa com o serviço do.

A política `AmazonEKSLocalOutpostClusterPolicy` inclui as seguintes permissões:

- **`ec2`**: permissões necessárias para que as instâncias do Amazon EC2 se integrem com êxito ao cluster como instâncias do ambiente de gerenciamento.
- **`ssm`**: permite a conexão do Amazon EC2 Systems Manager com a instância do ambiente de gerenciamento, que é usada pelo Amazon EKS para se comunicar e gerenciar o cluster local na sua conta.
- **`logs`**: permite que as instâncias enviem logs para o Amazon CloudWatch.
- **`secretsmanager`**: permite que as instâncias obtenham e excluam dados de bootstrap para as instâncias do ambiente de gerenciamento com segurança no AWS Secrets Manager.
- **`ecr`**: permite que Pods e contêineres em execução nas instâncias do ambiente de gerenciamento extraiam imagens de contêiner armazenadas no Amazon Elastic Container Registry.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSLocalOutpostClusterPolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada pela AWS: AmazonEKSLocalOutpostServiceRolePolicy

Não é possível anexar essa política a suas entidades do IAM. Quando você cria um cluster usando uma entidade principal do IAM que tem a permissão `iam:CreateServiceLinkedRole`, o Amazon EKS recria automaticamente a função vinculada ao serviço `AWSServiceRoleforAmazonEKSLocalOutpost` para você e anexa essa política a ela. Essa política permite que a função vinculada ao serviço chame produtos da AWS em seu nome para clusters locais.

A política `AmazonEKSLocalOutpostServiceRolePolicy` inclui as seguintes permissões:

- **`ec2`**: permite que o Amazon EKS trabalhe com segurança, com a rede e com outros recursos para iniciar e gerenciar com êxito instâncias do ambiente de gerenciamento na sua conta.
- **`ssm`**: permite a conexão do Amazon EC2 Systems Manager com a instância do ambiente de gerenciamento, que é usada pelo Amazon EKS para se comunicar e gerenciar o cluster local na sua conta.
- **`iam`**: permite que o Amazon EKS gerencie o perfil de instância associado às instâncias do ambiente de gerenciamento.
- **`secretsmanager`**: permite que o Amazon EKS coloque dados de bootstrap para as instâncias do ambiente de gerenciamento no AWS Secrets Manager para que ele possa ser referenciado com segurança durante o bootstrapping da instância.
- **`outposts`**: permite que o Amazon EKS obtenha informações do Outpost de sua conta para lançar com êxito um cluster local em um Outpost.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSLocalOutpostServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Atualizações do Amazon EKS para políticas gerenciadas pela AWS

Visualize detalhes sobre atualizações em políticas gerenciadas pela AWS para o Amazon EKS, desde que este serviço começou a monitorar essas alterações. Para obter alertas automáticos sobre

alterações feitas nesta página, inscreva-se no feed RSS na página Histórico de documentos do Amazon EKS.

Alteração	Descrição	Data
Permissões adicionadas para AWSServiceRoleForAmazonEKSNodegroup.	A permissão `ec2:DescribeCapacityReservations` foi adicionada para que o Amazon EKS possa descrever a reserva de capacidade na conta do usuário. A permissão `autoscaling:PutScheduledUpdateGroupAction` foi adicionada para permitir a configuração da escalabilidade programada em grupos de nós `CAPACITY_BLOCK` .	27 de junho de 2024
AmazonEKS_CNI_Policy: atualização para uma política existente	O Amazon EKS adicionou novas permissões `ec2:DescribeSubnets` para permitir que o Amazon VPC CNI plugin for Kubernetes veja a quantidade de endereços IP disponíveis em suas sub-redes da Amazon VPC. O plug-in CNI da VPC pode usar os endereços IP disponíveis em cada sub-rede para escolher as sub-redes com a maior quantidade de endereços IP disponíveis para usar ao criar uma interface de rede elástica.	4 de março de 2024
AmazonEKSWorkerNodePolicy: atualização para uma política existente	O Amazon EKS adicionou novas permissões para permitir EKS Pod Identities. O Amazon EKS Pod Identity Agent usa o perfil de nó.	26 de novembro de 2023

Alteração	Descrição	Data
Lançamento do AmazonEFSCSIDriverPolicy.	A AWS apresentou o `AmazonEFSCSIDriverPolicy` .	26 de julho de 2023
Permissões adicionadas para AmazonEKSClusterPolicy.	A permissão `ec2:DescribeAvailabilityZones` foi adicionada para permitir que o Amazon EKS obtenha os detalhes da AZ durante a descoberta automática de sub-redes ao criar balanceadores de carga.	7 de fevereiro de 2023
Condições da política atualizadas em Amazon BSCSIdriverPolicy.	Foram removidas as condições de política inválidas com caracteres curingas no campo de chave `StringLike` . Também foi adicionada uma nova condição `ec2:ResourceTag/kubernetes.io/created-for/pvc/name: "*"` a `ec2:DeleteVolume` , que permite que o driver da CSI do EBS exclua volumes criados pelo plug-in em árvore.	17 de novembro de 2022
Adicionadas permissões a AmazonEKSLocalOutpostServiceRolePolicy.	Adicionados `ec2:DescribeVPCAttribute` , `ec2:GetConsoleOutput` e `ec2:DescribeSecret` para permitir melhor validação de pré-requisitos e controle gerenciado do ciclo de vida. Também foram adicionados `ec2:DescribePlacementGroups` e `"arn:aws:ec2:*:*:placement-group/*"` a `ec2:RunInstances` para permitir controle de colocação das instâncias do Amazon EC2 do ambiente de gerenciamento no Outposts.	24 de outubro de 2022

Alteração	Descrição	Data
Atualize as permissões do Amazon Elastic Container Registry em AmazoneKS LocalOutpostClusterPolicy.	Movida a ação `ecr:GetDownloadUrlForLayer` de todas as seções de recurso para uma seção com escopo definido. Adicionado o recurso `arn:aws:ecr:*:*:repository/eks/*`. Removido o recurso `arn:aws:ecr:*:*:repository/eks/eks-certificates-controller-public`. Esse recurso é coberto pelo recurso `arn:aws:ecr:*:*:repository/eks/*` adicionado.	20 de outubro de 2022
Permissões adicionadas à AmazonEKSLocalOutpostClusterPolicy.	Adicionado o repositório do Amazon Elastic Container Registry `arn:aws:ecr:*:*:repository/kubelet-config-updater` para que as instâncias do ambiente de gerenciamento do cluster possam atualizar alguns argumentos `kubelet`.	31 de agosto de 2022
Apresentada a AmazonEKS LocalOutpostClusterPolicy.	A AWS apresentou o `AmazonEKSLocalOutpostClusterPolicy`.	24 de agosto de 2022
Apresentada a AmazonEKS LocalOutpostServiceRolePolicy.	A AWS apresentou o `AmazonEKSLocalOutpostServiceRolePolicy`.	23 de agosto de 2022
Introdução de AmazonEBS CSIDriverPolicy.	A AWS apresentou o `AmazonEBSCSIDriverPolicy`.	4 de abril de 2022

Alteração	Descrição	Data
Adição de permissões a AmazonEKSWorkerNodePolicy.	Adição de `ec2:DescribeInstanceTypes` para habilitar AMIs otimizadas para do Amazon EKS que podem detectar propriedades em nível de instâncias automaticamente.	21 de março de 2022
Permissões adicionadas para AWSServiceRoleForAmazonEKSNodegroup.	A permissão `autoscaling:EnableMetricsCollection` foi adicionada para que o Amazon EKS possa habilitar a coleta de métricas.	13 de dezembro de 2021
Permissões adicionadas para AmazonEKSClusterPolicy.	Permissões `ec2:DescribeAccountAttributes`, `ec2:DescribeAddresses` e `ec2:DescribeInternetGateways` adicionadas para permitir que o Amazon EKS crie uma função vinculada ao serviço para um Network Load Balancer.	17 de junho de 2021
O Amazon EKS passou a monitorar as alterações.	O Amazon EKS passou a controlar as alterações para as políticas gerenciadas da AWS.	17 de junho de 2021

Solução de problemas do IAM

Este tópico aborda alguns erros comuns que você pode encontrar ao usar o Amazon EKS com o IAM e como resolvê-los.

AccessDeniedException

Se você receber um `AccessDeniedException` ao chamar uma operação de API da AWS, é porque as credenciais da entidade principal do IAM que estão sendo usadas não têm as permissões necessárias para faze essa chamada.

```
An error occurred (AccessDeniedException) when calling the DescribeCluster operation:
User: arn:aws:iam::111122223333:user/user_name is not authorized to perform:
```

```
eks:DescribeCluster on resource: arn:aws:eks:region:111122223333:cluster/my-cluster
```

No exemplo de mensagem anterior, o usuário não tem permissões para chamar a operação da API `DescribeCluster` do Amazon EKS. Para conceder permissões de administrador do Amazon EKS a uma entidade principal do IAM, consulte Exemplos de políticas baseadas em identidade do Amazon EKS.

Para obter mais informações sobre o IAM, consulte Controlling access using policies (Controlar o acesso usando políticas) no Manual do usuário do IAM.

Não é possível ver Nodes (Nós) na guia Compute (Computação) ou qualquer coisa na guia Resources (Recursos), e você recebe um erro no AWS Management Console

Você pode ver uma mensagem de erro do console informando `Your current user or role does not have access to Kubernetes objects on this EKS cluster`. Verifique se o usuário da entidade principal do IAM com o qual você está usando o AWS Management Console tem as permissões necessárias. Para ter mais informações, consulte Permissões obrigatórias.

O aws-auth **`ConfigMap`** não concede acesso ao cluster

O AWS IAM Authenticator não permite um caminho no ARN da função usado no `ConfigMap`. Portanto, antes de especificar `rolearn`, remova o caminho. Por exemplo, altere `arn:aws:iam::111122223333:role/team/developers/eks-admin` para `arn:aws:iam::111122223333:role/eks-admin`.

Não estou autorizado a executar iam:PassRole

Se receber uma mensagem de erro informando que você não tem autorização para executar a ação `iam:PassRole`, suas políticas devem ser atualizadas para permitir a transmissão de um perfil ao Amazon EKS.

Alguns Serviços da AWS permitem que você passe um perfil existente para o serviço, em vez de criar um novo perfil de serviço ou perfil vinculado ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O erro exemplificado a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no Amazon EKS. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se você precisar de ajuda, entre em contato com seu administrador AWS. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha conta da AWS acessem meus recursos do Amazon EKS

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Amazon EKS oferece suporte a esses recursos, consulte Como o Amazon EKS funciona com o IAM.
- Para saber como conceder acesso a seus recursos em todas as Contas da AWS pertencentes a você, consulte Fornecimento de acesso a um usuário do IAM em outra Conta da AWS pertencente a você no Guia de usuário do IAM.
- Para saber como conceder acesso a seus recursos para terceiros Contas da AWS, consulte Fornecimento de acesso a Contas da AWS pertencentes a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Os contêineres do pod recebem o seguinte erro: **`An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`**

Os contêineres receberão esse erro se a aplicação estiver explicitamente fazendo solicitações ao endpoint global AWS STS (`https://sts.amazonaws`) e a conta de serviço do Kubernetes estiver configurada para utilizar um endpoint regional. É possível resolver o problema com uma das seguintes opções:

- Atualize o código da aplicação para remover chamadas explícitas para o endpoint global do AWS STS.
- Atualize o código da aplicação para fazer chamadas explícitas para endpoints regionais, como `https://sts.us-west-2.amazonaws.com`. Sua aplicação deve ter redundância incorporada para escolher uma Região da AWS diferente em caso de falha de serviço na Região da AWS. Para obter mais informações, consulte Gerenciar o AWS STS em uma Região da AWS no Guia do usuário do IAM.
- Configure as suas contas de serviço para utilizar o endpoint global. Todas as versões anteriores a `1.22` usavam o endpoint global por padrão, mas clusters da versão `1.22` e posteriores utilizam o endpoint regional por padrão. Para ter mais informações, consulte Configurar o endpoint do AWS Security Token Service para uma conta de serviço.

Funções e usuários padrão do Kubernetes criados pelo Amazon EKS

Quando você cria um cluster do Kubernetes, várias identidades padrão do Kubernetes são criadas nesse cluster para o funcionamento adequado do Kubernetes. O Amazon EKS cria identidades do Kubernetes para cada um de seus componentes padrão. As identidades fornecem controle de autorização baseada em perfil (RBAC) do Kubernetes para os componentes do cluster. Para mais informações, consulte Using RBAC Authorization (Usar autorização RBAC) na documentação do Kubernetes.

Quando você instala complementos opcionais no cluster, identidades adicionais do Kubernetes talvez sejam adicionadas a ele. Para obter mais informações sobre as identidades não abordadas neste tópico, consulte a documentação do complemento.

Você pode ver a lista de identidades do Kubernetes criadas pelo Amazon EKS no cluster usando o AWS Management Console ou ferramenta de linha de comando `kubectl`. Todas as identidades

de usuário aparecem nos logs de auditoria do `kube`, disponíveis a você por meio do Amazon CloudWatch.

AWS Management Console

Pré-requisito

A entidade principal do IAM que você usar deverá ter as permissões descritas em Permissões obrigatórias

Para visualizar as identidades criadas pelo Amazon EKS usando o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Na lista Clusters, escolha o cluster que contém as identidades do que você deseja visualizar.
3. Escolha a guia Recursos.
4. Em Resource types (Tipos de recursos), escolha Authorization (Autorização).
5. Escolha ClusterRoles, ClusterRoleBindings, Roles ou RoleBindings. Todos os recursos prefaciados com eks são criados pelo Amazon EKS. Outros recursos de identidade criados pelo Amazon EKS são:

   - O ClusterRole e o ClusterRoleBinding denominados aws-node. Os recursos aws-node são compatíveis com o Amazon VPC CNI plugin for Kubernetes, que o Amazon EKS instala em todos os clusters.
   - Um ClusterRole denominado vpc-resource-controller-role e um ClusterRoleBinding denominado vpc-resource-controller-rolebinding. Esses recursos são compatíveis com o controlador de recursos da Amazon VPC, que o Amazon EKS instala em todos os clusters.

   Além dos recursos que você vê no console, as seguintes identidades de usuário especiais existem no cluster, embora não estejam visíveis na sua configuração:

   - **`eks:cluster-bootstrap`**: usado para operações do `kubectl` durante a inicialização do cluster.
   - **`eks:support-engineer`**: usado para operações de gerenciamento do cluster.
6. Escolha um recurso específico para visualizar detalhes sobre ele. Por padrão, as informações são mostradas na visualização estruturada. No canto superior direito da página de detalhes, você pode escolher Raw view (Visualização bruta) para ver todas as informações sobre o recurso.

Kubectl

Pré-requisito

A entidade que você usa (AWS Identity and Access Management [IAM] ou OpenID Connect [OIDC]) para listar os recursos do Kubernetes no cluster deve ser autenticada pelo IAM ou pelo seu provedor de identidades OIDC. A entidade deve receber permissões para usar os verbos `get` e `list` do Kubernetes para os recursos `Role`, `ClusterRole`, `RoleBinding` e `ClusterRoleBinding` do cluster com que você deseja que a entidade trabalhe. Para obter mais informações sobre conceder às entidades do IAM acesso ao cluster, consulte the section called “Conceder acesso a APIs do Kubernetes”. Para obter mais informações sobre conceder às entidades autenticadas por seu próprio provedor OIDC acesso ao cluster, consulte Conceda aos usuários acesso ao Kubernetes com um provedor OIDC externo.

Para visualizar as identidades criadas pelo Amazon EKS usando o **`kubectl`**

Execute o comando para o tipo de recurso que você deseja ver. Todos os recursos retornados precedidos por eks são criados pelo Amazon EKS. Além dos recursos que retornados na saída dos comandos, as seguintes identidades de usuário especiais existem no cluster, embora não estejam visíveis na sua configuração:

- **`eks:cluster-bootstrap`**: usado para operações do `kubectl` durante a inicialização do cluster.
- **`eks:support-engineer`**: usado para operações de gerenciamento do cluster.

ClusterRoles: `ClusterRoles` estão no escopo do cluster, portanto, qualquer permissão concedida a um perfil se aplicará aos recursos em qualquer namespace do Kubernetes no cluster.

O comando a seguir retorna todos os `ClusterRoles` do Kubernetes criados pelo Amazon EKS no seu cluster.

```
kubectl get clusterroles | grep eks
```

Além dos `ClusterRoles` retornados na saída que são precedidos por eks, existem os `ClusterRoles` a seguir.

- **aws-node**: esse `ClusterRole` é compatível com o Amazon VPC CNI plugin for Kubernetes, que o Amazon EKS instala em todos os clusters.
- **vpc-resource-controller-role**: esse `ClusterRole` é compatível com o controlador de recursos Amazon VPC, que o Amazon EKS instala em todos os clusters.

Para ver a especificação de um `ClusterRole`, substitua *eks:k8s-metrics* no comando a seguir por um `ClusterRole` retornado na saída do comando anterior. O exemplo a seguir retorna a especificação para `ClusterRole` *eks:k8s-metrics*.

```
kubectl describe clusterrole eks:k8s-metrics
```

Veja um exemplo de saída abaixo.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names  Verbs
  ---------         -----------------  --------------  -----
                    [/metrics]         []              [get]
  endpoints         []                 []              [list]
  nodes             []                 []              [list]
  pods              []                 []              [list]
  deployments.apps  []                 []              [list]
```

ClusterRoleBindings: `ClusterRoleBindings` estão no escopo do cluster.

O comando a seguir retorna todos os `ClusterRoleBindings` do Kubernetes criados pelo Amazon EKS no seu cluster.

```
kubectl get clusterrolebindings | grep eks
```

Além dos `ClusterRoleBindings` retornados na saída, existem os `ClusterRoleBindings` a seguir.

- **aws-node**: esse `ClusterRoleBinding` é compatível com o Amazon VPC CNI plugin for Kubernetes, que o Amazon EKS instala em todos os clusters.
- **vpc-resource-controller-rolebinding**: esse `ClusterRoleBinding` é compatível com o controlador de recursos Amazon VPC, que o Amazon EKS instala em todos os clusters.

Para ver a especificação de um `ClusterRoleBinding`, substitua *`eks:k8s-metrics`* no comando a seguir por um `ClusterRoleBinding` retornado na saída do comando anterior. O exemplo a seguir retorna a especificação para `ClusterRoleBinding` *`eks:k8s-metrics`*.

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

Veja um exemplo de saída abaixo.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Roles: `Roles` estão no escopo de um namespace do Kubernetes. Todos os `Roles` criados pelo Amazon EKS estão no escopo do namespace do `kube-system`.

O comando a seguir retorna todos os `Roles` do Kubernetes criados pelo Amazon EKS no seu cluster.

```
kubectl get roles -n kube-system | grep eks
```

Para ver a especificação de um `Role`, substitua *`eks:k8s-metrics`* no comando a seguir pelo nome de um `Role` retornado na saída do comando anterior. O exemplo a seguir retorna a especificação para `Role` *`eks:k8s-metrics`*.

```
kubectl describe role eks:k8s-metrics -n kube-system
```

Veja um exemplo de saída abaixo.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names             Verbs
```

```
---------          -----------------  --------------              -----
daemonsets.apps    []                 [aws-node]                  [get]
deployments.apps   []                 [vpc-resource-controller]   [get]
```

RoleBindings: `RoleBindings` estão no escopo de um namespace do Kubernetes. Todos os `RoleBindings` criados pelo Amazon EKS estão no escopo do namespace do `kube-system`.

O comando a seguir retorna todos os `RoleBindings` do Kubernetes criados pelo Amazon EKS no seu cluster.

```
kubectl get rolebindings -n kube-system | grep eks
```

Para ver a especificação de um `RoleBinding`, substitua *`eks:k8s-metrics`* no comando a seguir por um `RoleBinding` retornado na saída do comando anterior. O exemplo a seguir retorna a especificação para `RoleBinding` *`eks:k8s-metrics`*.

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

Veja um exemplo de saída abaixo.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Validação de conformidade com o Amazon Elastic Kubernetes Service

Para saber se um Serviço da AWS está no escopo de programas de conformidade específicos, consulte Serviços da AWS no escopo por programa de conformidade e selecione o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de Conformidade da AWS.

É possível fazer download de relatórios de auditoria de terceiros usando o AWS Artifact. Para obter mais informações, consulte Baixar relatórios no AWS Artifact.

Sua responsabilidade de conformidade ao usar o Serviços da AWS é determinada pela confidencialidade dos seus dados, pelos objetivos de conformidade da sua empresa e pelos regulamentos e leis aplicáveis. A AWS fornece os seguintes recursos para ajudar com a conformidade:

- Guias de início rápido de segurança e conformidade: estes guias de implantação discutem considerações sobre arquitetura e fornecem as etapas para a implantação de ambientes de linha de base focados em segurança e conformidade na AWS.
- Arquitetura para segurança e conformidade com HIPAA no Amazon Web Services: esse whitepaper descreve como as empresas podem usar a AWS para criar aplicações adequadas aos padrões HIPAA.

  > **Note**
  >
  > Nem todos os Serviços da AWS estão qualificados pela HIPAA. Para mais informações, consulte a Referência dos serviços qualificados pela HIPAA.

- Recursos de Conformidade da AWS: essa coleção de manuais e guias pode ser aplicada ao seu setor e local.
- Guias de conformidade do cliente da AWS: entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as práticas recomendadas para proteção de Serviços da AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).
- Avaliar recursos com regras no Guia do desenvolvedor do AWS Config: o serviço AWS Config avalia como as configurações de recursos estão em conformidade com práticas internas, diretrizes do setor e regulamentos.
- AWS Security Hub: este Serviço da AWS fornece uma visão abrangente do seu estado de segurança na AWS. O Security Hub usa controles de segurança para avaliar os recursos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub.

- Amazon GuardDuty: este Serviço da AWS detecta possíveis ameaças às suas Contas da AWS, workloads, contêineres e dados ao monitorar o ambiente em busca de atividades suspeitas e maliciosas. O GuardDuty pode ajudar você a atender a diversos requisitos de conformidade, como o PCI DSS, com o cumprimento dos requisitos de detecção de intrusões requeridos por determinadas estruturas de conformidade.
- AWS Audit Manager: esse Serviço da AWS ajuda a auditar continuamente seu uso da AWS para simplificar a forma como você gerencia os riscos e a conformidade com regulamentos e padrões do setor.

Resiliência no Amazon EKS

A infraestrutura global da AWS se baseia em Regiões da AWS e zonas de disponibilidade. A Regiões da AWS oferece várias zonas de disponibilidade separadas e isoladas fisicamente que são conectadas com baixa latência, altas taxas de throughput e em redes altamente redundantes. Com as Zonas de Disponibilidade, você pode projetar e operar aplicativos e bancos de dados que executam o failover automaticamente entre as Zonas de Disponibilidade sem interrupção. As Zonas de Disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de datacenter tradicionais.

O Amazon EKS executa e escala ambientes de gerenciamento do Kubernetes em várias zonas de disponibilidade da AWS para garantir a alta disponibilidade. O Amazon EKS escala automaticamente instâncias do ambiente de gerenciamento baseadas em carga, detecta e substitui instâncias não íntegras do ambiente de gerenciamento e aplica patches automaticamente ao ambiente de gerenciamento. Depois de iniciar uma atualização de versão, o Amazon EKS atualiza seu ambiente de gerenciamento para você, mantendo a alta disponibilidade do ambiente de gerenciamento durante a atualização.

Este ambiente de gerenciamento consiste em pelo menos duas instâncias de servidor de API e três instâncias de `etcd` executadas em três zonas de disponibilidade em uma Região da AWS. Amazon EKS:

- Monitora ativamente a carga nas instâncias do plano de controle e as escala automaticamente para garantir alta performance.
- Detecta e substitui automaticamente instâncias não íntegras do ambiente de gerenciamento, reiniciando-as nas zonas de disponibilidade da Região da AWS, conforme a necessidade.
- Aproveita a arquitetura das Regiões da AWS, a fim de manter alta disponibilidade. Por isso, o Amazon EKS é capaz de oferecer um SLA para disponibilidade do endpoint do servidor de API.

Para obter mais informações sobre Regiões da AWS e Zonas de Disponibilidade, consulte Infraestrutura global da AWS.

Segurança da infraestrutura no Amazon EKS

Como um serviço gerenciado, o Amazon Elastic Kubernetes Service é protegido pela segurança de rede global da AWS. Para obter informações sobre serviços de segurança da AWS e como a AWS protege a infraestrutura, consulte Segurança na Nuvem AWS. Para projetar seu ambiente da AWS usando as práticas recomendadas de segurança da infraestrutura, consulte Proteção de Infraestrutura em Pilar de Segurança: AWS Well-Architected Framework.

Você usa chamadas à API publicadas pela AWS para acessar o Amazon EKS por meio da rede. Os clientes devem oferecer suporte para:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Ao criar um cluster do Amazon EKS, você especifica as sub-redes da VPC para uso do cluster. O Amazon EKS exige sub-redes em, pelo menos, duas zonas de disponibilidade. Recomendamos uma VPC com sub-redes públicas e privadas a fim de que o Kubernetes possa criar balanceadores de carga públicos nas sub-redes públicas que fazem o balanceamento de carga do tráfego para Pods em execução nos nós localizados em sub-redes privadas.

Para obter mais informações sobre as considerações da VPC, consulte Requisitos e considerações sobre a VPC e a sub-rede do Amazon EKS.

Se você criar a VPC e os grupos de nós com os modelos do AWS CloudFormation fornecidos na demonstração Começar a usar o Amazon EKS, o plano de controle e os grupos de segurança do nó serão definidos com as configurações recomendadas.

Para obter mais informações sobre considerações de grupos de segurança, consulte Considerações e requisitos sobre grupos de segurança do Amazon EKS.

Quando você cria um cluster, o Amazon EKS cria um endpoint para o servidor gerenciado de API do Kubernetes usado para se comunicar com o cluster (usando as ferramentas de gerenciamento do Kubernetes, como `kubectl`). Por padrão, esse endpoint do servidor de API é público para a Internet, e o acesso ao servidor de API é protegido por uma combinação do AWS Identity and Access Management (IAM) e do Controle de acesso com base em função (RBAC) nativo do Kubernetes.

Você pode habilitar o acesso privado ao servidor de API do Kubernetes para que todas as comunicações entre os nós e o servidor de API fiquem na VPC. Você pode limitar os endereços IP que podem acessar o servidor de API pela Internet ou desativar completamente o acesso à Internet para o servidor de API.

Para obter mais informações sobre como modificar o acesso do endpoint do cluster, consulte Modificar o acesso ao endpoint do cluster:

Você pode implementar políticas de rede do Kubernetes com o Amazon VPC CNI ou com ferramentas de terceiros, como o Project Calico. Para obter mais informações sobre o uso do Amazon VPC CNI para políticas de rede, consulte Configure seu cluster para as políticas de rede do Kubernetes. O Project Calico é um projeto de código-fonte aberto de terceiros. Para obter mais informações, consulte a documentação do Project Calico.

Análise de configuração e vulnerabilidade no Amazon EKS

A segurança é uma consideração essencial para configurar e manter clusters e aplicações do Kubernetes. A seguir são listados recursos para você analisar a configuração de segurança dos seus clusters do EKS, recursos para verificar vulnerabilidades e integrações com serviços da AWS que podem fazer essa análise para você.

Referência do Center for Internet Security (CIS) para Amazon EKS

A Referência do Center for Internet Security (CIS) para Kubernetes fornece orientação para configurações de segurança do Amazon EKS. O parâmetro de referência:

- É aplicável aos nós do Amazon EC2 (gerenciados e autogerenciados) nos quais você é responsável pelas configurações de segurança dos componentes do Kubernetes.
- Fornece uma maneira padrão aprovada pela comunidade de garantir que você configurou o cluster e nós do Kubernetes de modo seguro ao usar o Amazon EKS.
- Consiste em quatro seções: configuração de log de plano de controle, configurações de segurança do nó, políticas e serviços gerenciados.

- É compatível com todas as versões do Kubernetes atualmente disponíveis no Amazon EKS e pode ser executado usando kube-bench, uma ferramenta de código aberto padrão para verificar configuração usando o benchmark do CIS em clusters do Kubernetes.

Para saber mais, consulteApresentando o CIS Amazon EKS Benchmark.

Versões da plataforma do Amazon EKS

As versões da plataforma do Amazon EKS representam os recursos do ambiente de gerenciamento do cluster, incluindo quais sinalizadores do servidor de API do Kubernetes estão habilitados e a versão atual de patch do Kubernetes. Novos clusters são implantados com a versão mais recente da plataforma. Para obter detalhes, consulte Veja as versões da plataforma do Amazon EKS para cada versão do Kubernetes.

Você pode atualizar um cluster do Amazon EKS para versões mais recentes do Kubernetes. Conforme novas versões do Kubernetes são disponibilizadas no Amazon EKS, recomendamos que você atualize proativamente seus clusters para usarem a versão mais recente disponível. Para obter mais informações sobre versões do Kubernetes no EKS, consulte Compreeender o ciclo de vida da versão do Kubernetes no EKS.

Lista de vulnerabilidades do sistema operacional

Lista de vulnerabilidades do AL2023

Rastreie eventos de segurança ou privacidade para o Amazon Linux 2023 no Centro de segurança do Amazon Linux ou assine o feed RSS associado. Eventos de segurança e privacidade incluem uma visão geral do problema afetado, pacotes e instruções para atualizar suas instâncias para corrigir o problema.

Lista de vulnerabilidades do Amazon Linux 2

Rastreie eventos de segurança ou privacidade para o Amazon Linux 2 no Centro de segurança do Amazon Linux ou assine o feed RSS associado. Eventos de segurança e privacidade incluem uma visão geral do problema afetado, pacotes e instruções para atualizar suas instâncias para corrigir o problema.

Detecção de nós com o Amazon Inspector

É possível usar o Amazon Inspector para verificar a acessibilidade de rede não intencional dos nós e as vulnerabilidades nessas instâncias do Amazon EC2.

Detecção de clusters e nós com o Amazon GuardDuty

O Amazon GuardDuty é um serviço de detecção de ameaças que ajuda a proteger contas, contêineres, workloads e dados no ambiente da AWS. Entre outros recursos, o GuardDuty oferece os dois recursos a seguir que detectam possíveis ameaças aos seus clusters do EKS: Proteção do EKS e Monitoramento de runtime.

Para ter mais informações, consulte Detectar ameaças com o Amazon GuardDuty.

Práticas recomendadas de segurança para o Amazon EKS

As práticas recomendadas de segurança do Amazon EKS são mantidas no Github: https://aws.github.io/aws-eks-best-practices/security/docs

Política de segurança de pods

A controlador de admissão de política de segurança de Pod do Kubernetes valida a criação de Pod e atualiza as solicitações de acordo com um conjunto de regras. Por padrão, os clusters do Amazon EKS são fornecidos com uma política de segurança totalmente permissiva e sem restrições. Para obter mais informações, consulte Pod Security Policies (Políticas de segurança de pods) na documentação do Kubernetes.

> **Note**
>
> O `PodSecurityPolicy` (PSP) foi descontinuado no Kubernetes versão `1.21` e está programado para remoção no Kubernetes `1.25`. As PSPs estão sendo substituídas pelo Pod Security Admission (PSA), um controlador de admissão integrado que implementa os controles de segurança descritos nos padrões Pod Security Standards (PSS). Tanto o PSA quanto os PSS atingiram os estados de recursos beta e estão habilitados no Amazon EKS por padrão. Para resolver a remoção da PSP na versão `1.25`, recomendamos que você implemente PSS no Amazon EKS. Para obter mais informações, consulte Implementing Pod Security Standards in Amazon EKS (Implementando Pod Security Standards no Amazon EKS) no blog da AWS.

Política de segurança de Pod padrão do Amazon EKS

Os clusters do Amazon EKS com o Kubernetes versão `1.13` e posteriores têm uma política de segurança de Pod padrão denominada `eks.privileged`. Essa política não tem restrição em relação a qual tipo de Pod pode ser aceito no sistema, o que é equivalente a executar o Kubernetes com o controlador de `PodSecurityPolicy` desativado.

> **Note**
>
> Essa política foi criada para manter a compatibilidade com versões anteriores de clusters que não tinham o controlador `PodSecurityPolicy` habilitado. Você pode criar políticas mais restritivas para o seu cluster e para namespaces individuais e contas de serviço, e depois excluir a política padrão para habilitar as políticas mais restritivas.

Você pode visualizar a política padrão com o seguinte comando.

```
kubectl get psp eks.privileged
```

Veja um exemplo de saída abaixo.

```
NAME             PRIV   CAPS   SELINUX    RUNASUSER   FSGROUP    SUPGROUP
 READONLYROOTFS   VOLUMES
eks.privileged   true   *      RunAsAny   RunAsAny    RunAsAny   RunAsAny   false
      *
```

Para obter mais detalhes, você pode descrever a política com o comando a seguir.

```
kubectl describe psp eks.privileged
```

Veja um exemplo de saída abaixo.

```
Name:  eks.privileged

Settings:
  Allow Privileged:                       true
  Allow Privilege Escalation:             0xc0004ce5f8
  Default Add Capabilities:               <none>
  Required Drop Capabilities:             <none>
```

```
  Allowed Capabilities:                  *
  Allowed Volume Types:                  *
  Allow Host Network:                    true
  Allow Host Ports:                      0-65535
  Allow Host PID:                        true
  Allow Host IPC:                        true
  Read Only Root Filesystem:             false
  SELinux Context Strategy: RunAsAny
    User:                                <none>
    Role:                                <none>
    Type:                                <none>
    Level:                               <none>
  Run As User Strategy: RunAsAny
    Ranges:                              <none>
  FSGroup Strategy: RunAsAny
    Ranges:                              <none>
  Supplemental Groups Strategy: RunAsAny
    Ranges:                              <none>
```

Visualize o arquivo YAML completo para a política de segurança de Pod `eks.privileged`, a função de cluster e a vinculação de função do cluster em Instalar ou restaurar a política de segurança de Pod padrão.

Excluir política de segurança de Pod padrão do Amazon EKS

Se você criar políticas mais restritivas para os Pods, poderá, depois, excluir a política de segurança de Pod `eks.privileged` padrão do Amazon EKS para habilitar as políticas personalizadas.

> **Important**
>
> Se você estiver usando a versão `1.7.0` ou posterior do plug-in CNI e atribuir uma política de segurança personalizada do Pod à conta de serviço `aws-node` Kubernetes usada para os Pods `aws-node` implantados pelo Daemonset, essa política deverá ter `NET_ADMIN` em sua seção `allowedCapabilities` junto com `hostNetwork: true` e `privileged: true` na `spec` da política.

Para excluir a política de segurança de Pod padrão

1. Crie um arquivo chamado *`privileged-podsecuritypolicy.yaml`* com o conteúdo do arquivo de exemplo em Instalar ou restaurar a política de segurança de Pod padrão.

2. Exclua o YAML com o comando a seguir. Isso exclui a política de segurança de Pod padrão, a `ClusterRole` e a `ClusterRoleBinding` associadas a ela.

```
kubectl delete -f privileged-podsecuritypolicy.yaml
```

Instalar ou restaurar a política de segurança de Pod padrão

Se você estiver fazendo a atualização de uma versão anterior do Kubernetes ou se tiver modificado ou excluído a política de segurança de Pod `eks.privileged` padrão do Amazon EKS, poderá restaurá-la com as etapas a seguir.

Como instalar ou restaurar a política de segurança de Pod padrão

1. Crie um arquivo chamado *privileged-podsecuritypolicy.yaml* com o conteúdo a seguir.

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: eks.privileged
  annotations:
    kubernetes.io/description: 'privileged allows full unrestricted access to
      Pod features, as if the PodSecurityPolicy controller was not enabled.'
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  volumes:
  - '*'
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
```

```
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
  readOnlyRootFilesystem: false

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:podsecuritypolicy:privileged
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
rules:
- apiGroups:
  - policy
  resourceNames:
  - eks.privileged
  resources:
  - podsecuritypolicies
  verbs:
  - use

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:podsecuritypolicy:authenticated
  annotations:
    kubernetes.io/description: 'Allow all authenticated users to create privileged
 Pods.'
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:podsecuritypolicy:privileged
subjects:
  - kind: Group
```

```
    apiGroup: rbac.authorization.k8s.io
    name: system:authenticated
```

2. Aplique o YAML com o comando a seguir.

```
kubectl apply -f privileged-podsecuritypolicy.yaml
```

Perguntas frequentes sobre a remoção da política de segurança do Pod (PSP)

A `PodSecurityPolicy` foi descontinuada no Kubernetes`1.21` e removida no Kubernetes`1.25`. Se você estiver usando o PodSecurityPolicy no cluster, deverá migrar para os padrões de segurança Pod do Kubernetes integrados (PSS), ou para uma solução de política como código, antes de atualizar o cluster para a versão**`1.25`**, a fim de evitar interrupções nas workloads. Selecione qualquer pergunta frequente para saber mais.

O que é um PSP?

O PodSecurityPolicy é um controlador de admissão integrado que permite que um administrador de cluster controle os aspectos sensíveis à segurança da especificação do Pod. Se um Pod atender aos requisitos do PSP, o Pod será admitido no cluster como de costume. Se um Pod não atender aos requisitos de PSP, o Pod será rejeitado e não poderá ser executado.

A remoção da PSP é específica do Amazon EKS ou ela está sendo removida no upstream do Kubernetes?

Essa é uma mudança upstream no projeto do Kubernetes, e não uma mudança feita no Amazon EKS. O PSP foi descontinuado no Kubernetes `1.21` e removido no Kubernetes `1.25`. A comunidade Kubernetes identificou sérios problemas de usabilidade com o PSP. Isso incluiu a concessão acidental de permissões mais amplas do que o pretendido e a dificuldade de inspecionar quais PSPs se aplicam a uma determinada situação. Esses problemas não poderiam ser resolvidos sem fazer alterações significativas. Esse é o principal motivo pelo qual a comunidade Kubernetes decidiu remover o PSP.

Como verificar se estou usando PSPs em meus clusters do Amazon EKS?

Para verificar se você está usando PSPs em seu cluster, execute o seguinte comando:

```
kubectl get psp
```

Para ver os Pods que o PSPs do cluster está afetando, execute o seguinte comando: Esse comando gera o nome do Pod, o namespace e o PSPs:

```
kubectl get pod -A -o jsonpath='{range.items[?(@.metadata.annotations.kubernetes
\.io/psp)]}{.metadata.name}{"\t"}{.metadata.namespace}{"\t"}
{.metadata.annotations.kubernetes\.io/psp}{"\n"}'
```

Se eu estiver usando PSPs em meu cluster do Amazon EKS, o que posso fazer?

Antes de atualizar seu cluster para `1.25`, migre o PSPs para uma das seguintes alternativas:

- Kubernetes PSS.

- Soluções de política como código do ambiente do Kubernetes.

Em resposta à depreciação do PSP e à necessidade contínua de controlar a segurança do Pod desde o início, a comunidade Kubernetes criou uma solução integrada com o (PSS) e o Pod Security Admission (PSA). O webhook do PSA implementa os controles definidos no PSS.

Você pode revisar as melhores práticas para migrar PSPs para as PSS incorporadas no Guia de melhores práticas do EKS. Também recomendamos que você consulte nosso blog Implementing Pod Security Standards in Amazon EKS (Implementar padrões de segurança no Amazon EKS). Outras referências incluem Migrar do PodSecurityPolicy para o controlador de admissão PodSecurity integrado e o Mapeamento das PodSecurityPolicies para os padrões de segurança do Pod.

As soluções de política como código fornecem barreiras para orientar os usuários do cluster e evitam comportamentos indesejados por meio de controles automatizados prescritos. As soluções de política como código normalmente usam os Controladores de Admissão Dinâmicos do Kubernetes para interceptar o fluxo de solicitações do servidor da API Kubernetes usando uma chamada de webhook. As soluções de política como código alteram e validam as cargas de solicitação com base nas políticas escritas e armazenadas como código.

Existem várias soluções de política como código de código aberto disponíveis para o Kubernetes. Para analisar as melhores práticas para migrar PSPs para uma solução de política como código, consulte a seção Política como código da página de Segurança do Pod no GitHub.

Eu vejo uma PSP chamada **eks.privileged** no meu cluster. O que é e o que posso fazer sobre isso?

Os clusters do Amazon EKS com a versão `1.13` do Kubernetes e posteriores têm uma política de segurança de PSP padrão denominada `eks.privileged`. Essa política é criada em clusters `1.24` e anteriores. Não é usado no `1.25` e em clusters posteriores. O Amazon EKS migra automaticamente essa PSP para uma fiscalização baseada na PSS. Não é necessária nenhuma ação da sua parte.

O Amazon EKS fará alguma alteração nas PSPs já presentes em meu cluster existente quando eu atualizar meu cluster para a versão **1.25**?

Não. Além disso, `eks.privileged` que é uma PSP criada pelo Amazon EKS, nenhuma alteração será feita a outras PSPs no cluster quando você fizer o upgrade para o `1.25`.

O Amazon EKS impedirá uma atualização de cluster para a versão **1.25** se eu não tiver migrado da PSP?

Não. O Amazon EKS não impedirá a atualização do cluster para a versão `1.25` se você ainda não tiver migrado da PSP.

E se eu esquecer de migrar o PSPs para PSS/PSA ou para uma solução de política como código antes de atualizar meu cluster para a versão **1.25**? Posso migrar depois de atualizar meu cluster?

Quando um cluster que contém a PSP é atualizado para o Kubernetes versão `1.25`, o servidor da API não reconhece o recurso PSP na versão `1.25`. Isso pode fazer com que os Pods obtenham escopos de segurança incorretos. Para obter uma lista completa de implicações, consulte Migrar da PodSecurityPolicy para o controlador de admissão PodSecurity integrado.

Como essa alteração afeta a segurança do pod para cargas de trabalho do Windows?

Não esperamos nenhum impacto específico nas cargas de trabalho do Windows. PodSecurityContext tem um campo chamado `windowsOptions` na API `PodSpec v1` para Pods do Windows. Isso usa o PSS no Kubernetes `1.25`. Para obter mais informações e as práticas recomendadas para a imposição de PSS em workloads do Windows, consulte o Guia de práticas recomendadas do EKS e a documentação do Kubernetes.

Usar os segredos do AWS Secrets Manager com o Kubernetes

Para mostrar os segredos do Secrets Manager e os parâmetros do Parameter Store como arquivos montados nos Pods do Amazon EKS, você pode usar o AWS Secrets and Configuration Provider (ASCP) para o Driver de CSI do Kubernetes Secrets Store.

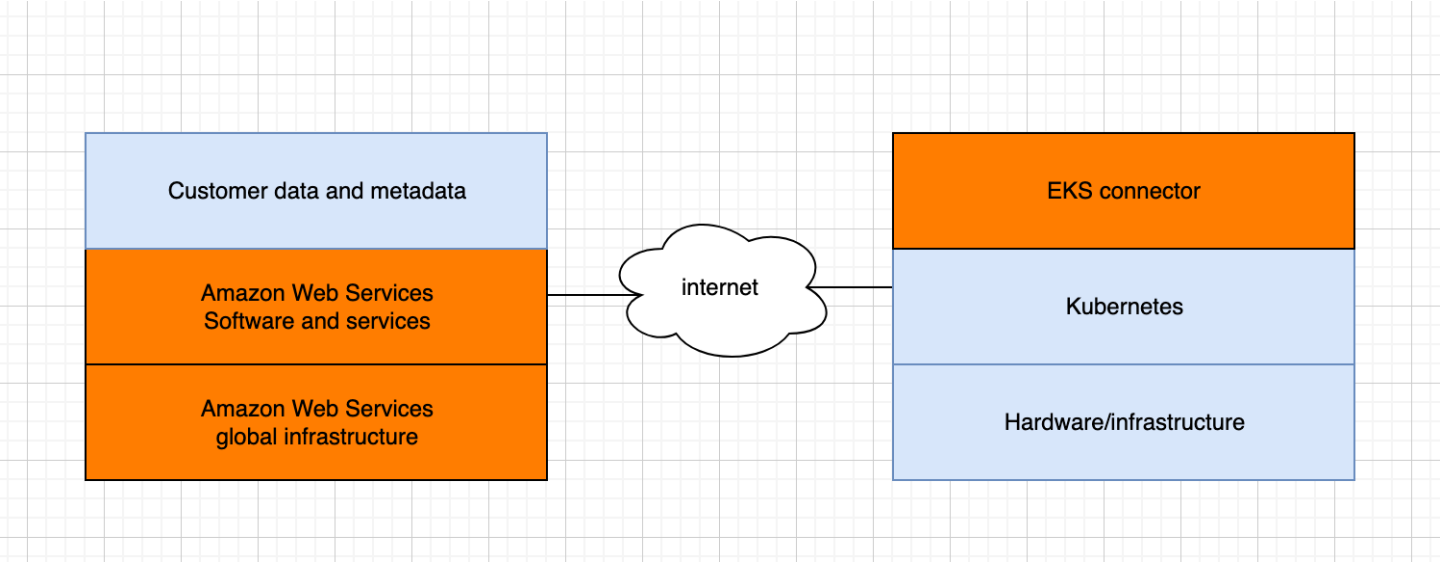
Com o ASCP, você pode armazenar e gerenciar segredos no Secrets Manager e, em seguida, recuperá-los por meio de workloads executadas no Amazon EKS. Você pode usar perfis do IAM e políticas do IAM para limitar o acesso aos segredos a determinados Pods do Kubernetes em um cluster. O ASCP recupera a identidade do Pod e troca a identidade por um perfil do IAM. O ASCP assume o perfil do IAM do Pod e, assim, consegue recuperar segredos do Secrets Manager autorizados para esse perfil.

Se você usar a alternância automática do Secrets Manager para seus segredos, também poderá usar o recurso de reconciliador de alternância do driver da CSI do armazenamento de segredos para garantir que está recuperando o segredo mais recente do Secrets Manager.

Para obter mais informações, consulte Uso dos segredos do Secrets Manager o Amazon EKS no Guia do usuário do AWS Secrets Manager.

Considerações sobre o Amazon EKS Connector

O Amazon EKS Connector é um componente de código aberto que é executado no cluster do Kubernetes. Esse cluster pode estar localizado fora do ambiente AWS. Isso gera considerações adicionais para responsabilidades de segurança. Essa configuração pode ser ilustrada pelo diagrama a seguir. A cor laranja representa responsabilidades da AWS, enquanto a cor azul representa as responsabilidades do cliente:



Este tópico descreve as diferenças no modelo de responsabilidade quando o cluster conectado está fora da AWS.

Responsabilidades da AWS

- Manutenção, compilação e entrega do Amazon EKS Connector, um componente de código aberto executado no cluster do Kubernetes de um cliente e que se comunica com a AWS.
- Manutenção da segurança de comunicação da camada de transporte e aplicação entre o cluster do Kubernetes conectado e os serviços da AWS.

Responsabilidades do cliente

- Segurança específica do cluster do Kubernetes, especificamente nas seguintes linhas:
  - Os segredos do Kubernetes devem ser criptografados e protegidos adequadamente.
  - Bloqueio do acesso ao namespace `eks-connector`.
- Configuração de permissões para controle de acesso baseado em funções (RBAC) a fim de gerenciar o acesso das entidades principais do IAM da AWS. Para obter instruções, consulte Conceder acesso para visualizar recursos de clusters do Kubernetes em um console do Amazon EKS.
- Instalação e upgrade do Amazon EKS Connector.
- Manutenção do hardware, software e infraestrutura de suporte do cluster do Kubernetesconectado.
- Proteção de suas contas da AWS (por exemplo, protegendo as credenciais de usuário raiz).

Visualizar os recursos do Kubernetes

É possível visualizar os recursos do Kubernetes implantados no cluster usando o AWS Management Console. Não é possível visualizar os recursos do Kubernetes com a AWS CLI ou o `eksctl`. Para visualizar os recursos do Kubernetes utilizando uma ferramenta de linha de comando, use `kubectl`.

Pré-requisito

Para visualizar a guia Recursos e a seção Nós na guia Computação no AWS Management Console, a entidade principal do IAM que você está usando deve ter permissões específicas do IAM e do Kubernetes. Para ter mais informações, consulte Permissões obrigatórias.

Como visualizar os recursos do Kubernetes com o AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Na lista Clusters, escolha o cluster que contém os recursos do Kubernetes que você deseja visualizar.
3. Selecione a guia Recursos.
4. Selecione um grupo Resource type (Tipo de recurso) cujos recursos você deseja visualizar, como Workloads. Você visualizará uma lista de tipos de recursos nesse grupo.
5. Selecione um tipo de recurso, como Deployments (Implantações), no grupo Workloads. Você verá uma descrição do tipo de recurso, um link para a documentação do Kubernetes para obter mais informações sobre esse tipo de recurso, bem como uma lista dos recursos desse tipo implantados no cluster. Se essa lista estiver vazia, significa que não há recursos desse tipo implantados no cluster.
6. Selecione um recurso para exibir mais informações sobre ele. Teste os exemplos a seguir:

    - Selecione o grupo Workloads, escolha o tipo de recurso Deployments (Implantações) e depois selecione o recurso coredns. Por padrão, ao selecionar um recurso, você está em Structured view (Visualização estruturada). Para alguns tipos de recursos, há uma seção Pods em Structured view (Visualização estruturada). Esta seção mostra os Pods gerenciados pela workload. É possível selecionar qualquer Pod listado para visualizar informações sobre o Pod. Nem todos os tipos de recursos mostram informações em Structured view (Visualização estruturada). Se a opção Raw view (Visualização bruta) for selecionada no canto superior direito da página do recurso, será possível ver a resposta JSON completa da API do Kubernetes para esse recurso.

- Selecione o grupo Cluster e depois o tipo de recurso Nodes (Nós). Você verá uma lista com todos os nós do seu cluster. Esses nós podem ser de qualquer Tipo de nó do Amazon EKS. Esta é a mesma lista visível na seção Nodes (Nós) ao selecionar a guia Compute (Computação) do seu cluster. Escolha um recurso de nó da lista. Em Structured view (Visualização estruturada), também é possível ver uma seção Pods. Essa seção mostra todos os Pods em execução no nó.

Permissões obrigatórias

Para visualizar a guia Recursos e a seção Nós na guia Computação no AWS Management Console, a entidade principal do IAM que você está usando deve ter permissões específicas mínimas do IAM e do Kubernetes. Conclua as seguintes etapas para atribuir as permissões necessárias às entidades principais do IAM.

1. Verifique se `eks:AccessKubernetesApi` e as outras permissões necessárias do IAM para visualizar os recursos do Kubernetes estão atribuídas à entidade principal do IAM que você está usando. Para obter mais informações sobre como editar permissões para uma entidade principal do IAM, consulte Controle de acesso das entidades principais no Guia do usuário do IAM. Para saber mais sobre como editar as permissões de um perfil, consulte Modificar a política de permissões de um perfil (console), no Guia do usuário do IAM.

   O seguinte exemplo de política inclui as permissões necessárias para uma entidade principal visualizar recursos do Kubernetes para todos os clusters da conta. Substitua *111122223333* pelo ID de sua conta da AWS.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "eks:ListFargateProfiles",
                   "eks:DescribeNodegroup",
                   "eks:ListNodegroups",
                   "eks:ListUpdates",
                   "eks:AccessKubernetesApi",
                   "eks:ListAddons",
                   "eks:DescribeCluster",
                   "eks:DescribeAddonVersions",
   ```

```
                "eks:ListClusters",
                "eks:ListIdentityProviderConfigs",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:GetParameter",
            "Resource": "arn:aws:ssm:*:111122223333:parameter/*"
        }
    ]
}
```

Para visualizar nós em clusters conectados, o perfil do IAM do conector do Amazon EKS deve ser capaz de representar a entidade principal no cluster. Isso permite que o Conecte um cluster do Kubernetes a um console de gerenciamento do Amazon EKS com o Amazon EKS Connector mapeie a entidade principal para um usuário do Kubernetes.

2. Crie uma `rolebinding` ou `clusterrolebinding` do Kubernetes que esteja vinculada a uma `role` ou `clusterrole` do Kubernetes que tenha as permissões necessárias para visualizar os recursos do Kubernetes. Para saber mais sobre funções e associações de função do Kubernetes, consulte Using RBAC Authorization (Usar a autorização RBAC), na documentação do Kubernetes. É possível aplicar ao cluster um dos seguintes manifestos que criam uma `role` e `rolebinding` ou uma `clusterrole` e `clusterrolebinding` com as permissões necessárias do Kubernetes:

   Visualizar recursos do Kubernetes em todos os namespaces

   O nome do grupo no arquivo é `eks-console-dashboard-full-access-group`. Aplique o manifesto ao seu cluster com o seguinte comando:

   ```
   kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
   ```

   Visualizar recursos do Kubernetes em um namespace específico

   O namespace neste arquivo é `default`. O nome do grupo no arquivo é `eks-console-dashboard-restricted-access-group`. Aplique o manifesto ao seu cluster com o seguinte comando:

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
```

Se for necessário modificar o nome do grupo, o namespace, as permissões do Kubernetes ou qualquer outra configuração no arquivo, baixe esse arquivo e edite-o antes de aplicá-lo ao cluster:

1. Baixe o arquivo com um dos seguintes comandos:

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
```

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
```

2. Edite esse arquivo conforme for necessário.
3. Aplique o manifesto ao seu cluster com um dos seguintes comandos:

```
kubectl apply -f eks-console-full-access.yaml
```

```
kubectl apply -f eks-console-restricted-access.yaml
```

3. Mapeie a entidade principal do IAM para o usuário ou grupo do Kubernetes no `aws-auth` `ConfigMap`. É possível usar uma ferramenta como o `eksctl` para atualizar o `ConfigMap`, ou você pode atualizá-lo manualmente via edição.

> **Important**
>
> Convém usar `eksctl` ou outra ferramenta para editar o `ConfigMap`. Para saber mais sobre outras ferramentas que podem ser usadas, consulte Usar ferramentas para fazer alterações em `aws-auth` `ConfigMap` nos guias de práticas recomendadas do Amazon EKS. Um `aws-auth` `ConfigMap` formatado incorretamente pode fazer com que você perca o acesso ao cluster.

eksctl

Pré-requisito

Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

1. Visualize os mapeamentos atuais no `ConfigMap`. Substitua o *`my-cluster`* pelo nome do cluster. Substitua *`region-code`* pela Região da AWS em que está o cluster.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

   Veja um exemplo de saída abaixo.

```
ARN
                  USERNAME                                GROUPS
            ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

2. Adicione um mapeamento referente a uma função. Esse exemplo pressupõe que você tenha anexado as permissões do IAM na primeira etapa a uma função denominada *`my-console-viewer-role`*. Substitua *`111122223333`* pelo ID da sua conta.

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:role/my-console-viewer-role \
    --group eks-console-dashboard-full-access-group \
    --no-duplicate-arns
```

   Important

   O ARN da função não pode incluir um caminho, como `role/my-team/developers/my-role`. O formato do ARN deve ser

`arn:aws:iam::111122223333:role/my-role`. Neste exemplo, `my-team/developers/` precisa ser removido.

Veja um exemplo de saída abaixo.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "arn:aws:iam::111122223333:role/my-console-viewer-role" to auth ConfigMap
```

3. Adicione um mapeamento referente a um usuário. As melhores práticas do IAM recomendam que você conceda permissões para perfis e não para usuários. Esse exemplo pressupõe que você tenha anexado as permissões do IAM na primeira etapa a um usuário denominado *my-user*. Substitua *111122223333* pelo ID da sua conta.

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user \
    --group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

Veja um exemplo de saída abaixo.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws:iam::111122223333:user/my-user" to auth ConfigMap
```

4. Visualize os mapeamentos no `ConfigMap` novamente.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

Veja um exemplo de saída abaixo.

```
ARN
                    USERNAME                                GROUPS
                      ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

```
arn:aws:iam::111122223333:role/my-console-viewer-role
                                                        eks-console-
dashboard-full-access-group
arn:aws:iam::111122223333:user/my-user
                                                        eks-console-
dashboard-restricted-access-group
```

Edit ConfigMap manually

Para obter mais informações sobre como adicionar usuários ou perfis do IAM ao `aws-auth` `ConfigMap`, consulte Adicione um usuário do IAM ao cluster do Amazon EKS.

1. Abra o `ConfigMap` de `aws-auth` para edição.

```
kubectl edit -n kube-system configmap/aws-auth
```

2. Adicione os mapeamentos ao `aws-auth` `ConfigMap`, mas não substitua os mapeamentos existentes. O seguinte exemplo adiciona mapeamentos entre as entidades principais do IAM com permissões adicionadas na primeira etapa e os grupos do Kubernetes criados na etapa anterior:
   - A função *my-console-viewer-role* e o `eks-console-dashboard-full-access-group`.
   - O usuário *my-user* e o `eks-console-dashboard-restricted-access-group`.

   Esses exemplos pressupõem que você tenha anexado as permissões do IAM da primeira etapa a um perfil denominado *my-console-viewer-role* e a um usuário denominado *my-user*. Substitua *111122223333* pelo ID de sua conta da AWS.

```
apiVersion: v1
data:
mapRoles: |
  - groups:
    - eks-console-dashboard-full-access-group
    rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
    username: my-console-viewer-role
mapUsers: |
  - groups:
    - eks-console-dashboard-restricted-access-group
    userarn: arn:aws:iam::111122223333:user/my-user
```

```
    username: my-user
```

> **Important**
>
> O ARN da função não pode incluir um caminho, como `role/my-team/developers/my-console-viewer-role`. O formato do ARN deve ser `arn:aws:iam::111122223333:role/my-console-viewer-role`. Neste exemplo, `my-team/developers/` precisa ser removido.

3. Salve o arquivo e saia do seu editor de texto.

Monitorar a performance de clusters e visualizara logs

É possível observar seus dados no Amazon EKS usando várias ferramentas de monitoramento ou registro em log disponíveis. Os dados de log do Amazon EKS podem ser transmitidos para os Serviços da AWS ou para ferramentas de parceiros para análise de dados. Existem muitos serviços disponíveis no AWS Management Console que fornecem dados para solucionar problemas do Amazon EKS. Você também pode usar uma solução de código aberto compatível com a AWS para monitorar a infraestrutura do Amazon EKS.

Depois de selecionar Clusters no painel de navegação esquerdo do console do Amazon EKS, você pode visualizar a integridade e os detalhes do cluster selecionando o nome do cluster. Para visualizar detalhes sobre todos os recursos existentes do Kubernetes que estão implantados no cluster, consulte Visualizar os recursos do Kubernetes.

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e a performance do Amazon EKS e das soluções da AWS. Recomendamos coletar dados de monitoramento de todas as partes da sua solução da AWS. Dessa maneira, é possível depurar uma falha de vários pontos com facilidade, caso ocorra. Antes de começar a monitorar o Amazon EKS, certifique-se de que seu plano de monitoramento aborde as seguintes questões.

- Quais são as suas metas? Você precisa de notificações em tempo real se os seus clusters forem escalados drasticamente?
- Que recursos precisam ser observados?
- Com que frequência esses recursos precisam ser observados? Sua empresa deseja responder rapidamente a riscos?
- Que ferramentas você pretende usar? Se você já executa o AWS Fargate como parte da inicialização, poderá usar o roteador de logs incorporado.
- Quem você pretende designar para realizar as tarefas de monitoramento?
- Para quem você deseja que notificações sejam enviadas quando algo der errado?

Registrar em log e monitorar no Amazon EKS

O Amazon EKS fornece ferramentas incorporadas para registro em log e monitoramento. Os logs do ambiente de gerenciamento registram todas as chamadas de API para os clusters, as informações de auditoria que capturam quais usuários realizaram quais ações nesses clusters, bem como

informações baseadas em perfil. Para obter mais informações, consulte Registro e monitoramento no Amazon EKS na Orientação prescritiva da AWS.

O registro em log do plano de controle do Amazon EKS fornece logs de auditoria e diagnóstico diretamente do plano de controle do Amazon EKS para o CloudWatch Logs em sua conta. Esses logs facilitam a proteção e a execução dos clusters. Você pode selecionar os tipos exatos de logs que precisa, e eles serão enviados como fluxos de log para cada cluster do Amazon EKS no CloudWatch. Para obter mais informações, consulte Enviar logs do ambiente de gerenciamento para o CloudWatch Logs.

> **Note**
>
> Ao verificar os logs do autenticador do Amazon EKS no Amazon CloudWatch, são exibidas as entradas que contêm texto semelhante ao exemplo a seguir.
>
> ```
> level=info msg="mapping IAM role" groups="[]"
>  role="arn:aws:iam::111122223333:role/XXXXXXXXXXXXXXXXXX-
> NodeManagerRole-XXXXXXXX" username="eks:node-manager"
> ```
>
> As entradas que contenham esse texto são esperadas. O `username` é uma função de serviço interna do Amazon EKS que executa operações específicas para grupos de nós gerenciados e o Fargate.
> Para registro em log personalizável de baixo nível, está disponível o registro em log do Kubernetes.

O Amazon EKS é integrado ao AWS CloudTrail, um serviço que fornece um registro das ações tomadas por um usuário, uma função ou um serviço da AWS no Amazon EKS. O CloudTrail captura todas as chamadas de API para o Amazon EKS como eventos. As chamadas capturadas incluem as chamadas do console do Amazon EKS e as chamadas de código para as operações da API do Amazon EKS. Para obter mais informações, consulte Registrar chamadas de API em log como eventos do AWS CloudTrail.

O servidor de API do Kubernetes expões várias métricas úteis para monitoramento e análise. Para obter mais informações, consulte Monitorar métricas do cluster com o Prometheus.

Para configurar o Fluent Bit para logs personalizados do Amazon CloudWatch, consulte Configurar o Fluent Bit, no Guia do usuário do Amazon CloudWatch.

Ferramentas do Amazon EKS de registro em log e monitoramento

A Amazon Web Services fornece várias ferramentas que você pode usar para monitorar o Amazon EKS. É possível configurar algumas ferramentas para definir o monitoramento automático, mas algumas exigem chamadas manuais. Convém automatizar as tarefas de monitoramento o tanto quanto permitido pelo seu ambiente e conjunto de ferramentas.

Ferramentas de registro em log

Áreas	Ferramenta	Descrição	Configuração
Aplicações	Amazon CloudWatch Container Insights	Ele coleta, agrega e resume métricas e logs de seus aplicativos e microsserviços em contêineres.	Procedimento de configuração
Ambiente de gerenciamento	AWS CloudTrail	Ele registra chamadas de API por parte de um usuário, uma função ou um serviço.	Procedimento de configuração
Várias áreas para instâncias do AWS Fargate	Roteador de log do AWS Fargate	Para instâncias do AWS Fargate, ele transmite logs para os serviços da AWS ou ferramentas de parceiros. Utiliza o AWS for Fluent Bit. Os logs podem ser transmitidos para outros Serviços	Procedimento de configuração

Áreas	Ferramenta	Descrição	Configuração
		da AWS ou para ferramentas de parceiros.	

Ferramentas de monitoramento

Áreas	Ferramenta	Descrição	Configuração
Aplicações	CloudWatch Container Insights	O CloudWatch Container Insights coleta, agrega e resume métricas e logs das suas aplicações e microsserviços conteinerizados.	Procedimento de configuração
Aplicações	AWS Distro for OpenTelemetry (ADOT)	Ele coleta e envia métricas correlacionadas, dados de rastreamento e metadados para serviços de monitoramento da AWS ou parceiros. Ele pode ser configurado por meio do CloudWatch Container Insights.	Procedimento de configuração

Áreas	Ferramenta	Descrição	Configuração
Aplicações	Amazon DevOps Guru	Ele detecta a performance e a disponibilidade operacionais no nível do nó.	Procedimento de configuração
Aplicações	AWS X-Ray	Ele recebe dados de rastreamento sobre a sua aplicação. Esses dados de rastreamento incluem solicitaç ões e metadados de entrada e de saída sobre as solicitaç ões. Para o Amazon EKS, a implement ação requer o complemento de OpenTelemetry.	Procedimento de configuração

Áreas	Ferramenta	Descrição	Configuração
Aplicações	Amazon CloudWatch Observability Operator	O Amazon CloudWatch Observability Operator coleta métricas, logs e dados de rastreamento. Ele os envia para o Amazon CloudWatch e o AWS X-Ray.	Procedimento de configuração
Aplicações/ambiente de gerenciamento	Prometheus	O Prometheus pode ser usado para monitorar métricas e alertas para aplicações e ambiente de gerenciamento.	Procedimento de configuração

Monitorar métricas do cluster com o Prometheus

O Prometheus é um banco de dados de monitoramento e séries temporais que extrai endpoints. Ele permite consultar, agregar e armazenar dados coletados. Você também pode usá-lo para alertas e agregação de alertas. Este tópico explica como configurar o Prometheus como uma opção gerenciada ou de código aberto. Monitorar o ambiente de gerenciamento de métricas do Amazon EKS é um caso de uso comum.

O Amazon Managed Service for Prometheus é um serviço de monitoramento e emissão de alertas compatível com o Prometheus que facilita o monitoramento de aplicações conteinerizadas e infraestrutura em escala. É um serviço totalmente gerenciado que dimensiona automaticamente a ingestão, o armazenamento, a consulta e o alerta de métricas. Também se integra aos serviços de segurança da AWS para permitir acesso rápido e seguro aos dados. É possível usar a linguagem

de consulta PromQL de código aberto para consultar suas métricas e emitir alertas sobre elas. Você também pode usar o gerenciador de alertas no Amazon Managed Service for Prometheus para configurar regras de alertas para alertas críticos. Você pode então enviar esses alertas críticos para um tópico do Amazon SNS.

Para obter mais informações sobre como usar as métricas do Prometheus depois de ativá-las, consulte o Guia do usuário do Amazon Managed Service for Prometheus.

Há várias opções diferentes para usar o Prometheus com o Amazon EKS:

- Você pode ativar as métricas do Prometheus ao criar pela primeira vez um cluster Amazon EKS, o que é abordado neste tópico.
- Se você já tem um cluster do Amazon EKS existente, pode criar seu próprio extrator do Prometheus. Para obter mais informações, consulte Criar um extrator no Guia do usuário do Amazon Managed Service for Prometheus.
- É possível implantar o Prometheus via Helm. Para obter mais informações, consulte Implementar o Prometheus usando o Helm.
- Você pode visualizar as métricas brutas do ambiente de gerenciamento no formato Prometheus. Para obter mais informações, consulte Visualizar as métricas brutas do ambiente de gerenciamento no formato Prometheus.

Etapa 1: ativar as métricas do Prometheus ao criar um cluster

> **Important**
>
> Os recursos do Amazon Managed Service for Prometheus estão fora do ciclo de vida do cluster e precisam ser mantidos separados do cluster. Ao excluir seu cluster, certifique-se de excluir também todos os extratores relevantes para interromper os custos aplicáveis. Para obter mais informações, consulte Encontrar e excluir extratores no Guia do usuário do Amazon Managed Service for Prometheus.

Quando você cria um novo cluster, é possível ativar a opção de enviar métricas para o Prometheus. No AWS Management Console, essa opção está disponível na etapa Configurar observabilidade da criação de um novo cluster. Para obter mais informações, consulte Criar um cluster do Amazon EKS..

O Prometheus descobre e coleta métricas do seu cluster por meio de um modelo baseado em pull chamado extração. Os extratores são configurados para coletar dados de sua infraestrutura de cluster e aplicações conteinerizadas.

Quando você ativa a opção de enviar métricas do Prometheus, o Amazon Managed Service for Prometheus fornece um extrator sem agente totalmente gerenciado. Use as seguintes opções de Configuração avançada para personalizar o extrator padrão conforme necessário.

Alias do extrator

(Opcional) Insira um alias exclusivo para o extrator.

Destination (Destino)

Escolha um espaço de trabalho do Amazon Managed Service for Prometheus. Um espaço de trabalho é um espaço lógico dedicado ao armazenamento e à consulta de métricas do Prometheus. Com esse espaço de trabalho, você poderá visualizar métricas do Prometheus em todas as contas que têm acesso a ele. A opção Criar novo espaço de trabalho instrui o Amazon EKS a criar um espaço de trabalho em seu nome usando o Alias de espaço de trabalho fornecido por você. Com a opção Selecionar espaço de trabalho existente, você pode selecionar um espaço de trabalho existente em uma lista suspensa. Para obter mais informações sobre espaços de trabalho, consulte Gerenciamento de espaços de trabalho no Guia do usuário do Amazon Managed Service for Prometheus.

Acesso ao serviço

Esta seção resume as permissões que você concede ao enviar métricas do Prometheus:

- Permita que o Amazon Managed Service for Prometheus descreva o cluster do Amazon EKS extraído
- Permitir gravação remota no espaço de trabalho do Amazon Managed Service para Prometheus

Se o `AmazonManagedScraperRole` já existir, o extrator o usará. Escolha o link `AmazonManagedScraperRole` para ver os Detalhes da permissão. Se ainda não houver um `AmazonManagedScraperRole`, escolha o link de detalhes Visualizar permissão para ver as permissões específicas que você está concedendo enviando métricas do Prometheus.

Sub-redes

Visualize as sub-redes que o extrator herdará. Se você precisar alterá-los, volte para a etapa cluster Especificar rede da criação do cluster.

Grupos de segurança

Visualize os grupos de segurança que o extrator herdará. Se você precisar alterá-los, volte para a etapa cluster Especificar rede da criação do cluster.

Configuração do extrator

Modifique a configuração do extrator no formato YAML conforme necessário. Para isso, use o formulário ou faça upload de um arquivo YAML substituto. Para obter mais informações, consulte Configuração do extrator no Guia do usuário do Amazon Managed Service for Prometheus.

O Amazon Managed Service for Prometheus se refere ao extrator sem agente criado com o cluster como um coletor gerenciado pela AWS. Para obter mais informações sobre coletores gerenciados pela AWS, consulte coletores gerenciados pela AWS no Guia do usuário do Amazon Managed Service for Prometheus.

Important

Você deve configurar o `aws-auth ConfigMap` para conceder permissões no cluster ao extrator. Para obter mais informações, consulte Configurar o cluster do Amazon EKS no Guia do usuário do Amazon Managed Service for Prometheus.

Etapa 2: visualizar detalhes do extrator do Prometheus

Depois de criar um cluster com a opção de métricas do Prometheus ativada, você poderá visualizar os detalhes do seu extrator do Prometheus. Ao visualizar seu cluster no AWS Management Console, escolha a guia Observabilidade. Uma tabela mostra uma lista de extratores para o cluster, incluindo informações como ID, alias, status e data de criação do extrator.

Para ver mais detalhes sobre o extrator, escolha um link de ID do extrator. Por exemplo, você também pode ver a configuração do extrator, o nome do recurso da Amazon (ARN), o URL de gravação remota e as informações de rede. É possível usar o ID do extrator como entrada para as operações de API do Amazon Managed Service for Prometheus como `DescribeScraper` e `DeleteScraper`. Também é possível usar a API para criar mais extratores.

Para obter mais informações sobre como usar a API Prometheus, consulte a Referência da API do Amazon Managed Service for Prometheus.

Implementar o Prometheus usando o Helm

Como alternativa ao uso do Amazon Managed Service for Prometheus, é possível implantar o Prometheus em seu cluster com o Helm V3. Se você já instalou o Helm, poderá verificar sua versão com o comando `helm version`. O Helm é um gerenciador de pacotes para clusters do Kubernetes. Para obter mais informações sobre o Helm e como instalá-lo, consulte Implantar aplicações com o Helm no Amazon EKS.

Após configurar o Helm para o cluster do Amazon EKS, você poderá usá-lo para implantar o Prometheus com as etapas a seguir.

Para implantar o Prometheus usando o Helm

1. Crie um namespace Prometheus.

```
kubectl create namespace prometheus
```

2. Adicione o gráfico do repositório do `prometheus-community`.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

3. Implante o Prometheus.

```
helm upgrade -i prometheus prometheus-community/prometheus \
    --namespace prometheus \
    --set alertmanager.persistence.storageClass="gp2" \
    --set server.persistentVolume.storageClass="gp2"
```

> **Note**
>
> Se você receber o erro `Error: failed to download "stable/prometheus" (hint: running `helm repo update` may help)` ao executar este comando, execute `helm repo update prometheus-community` e, em seguida, tente executar o comando da etapa 2 novamente.
> Se você receber o erro `Error: rendered manifests contain a resource that already exists` ao executar este comando, execute `helm uninstall` *`your-release-name`* `-n` *`namespace`* e, em seguida, tente executar o comando da etapa 3 novamente.

4. Verifique se todos os Pods no namespace `prometheus` estão no estado `READY`.

```
kubectl get pods -n prometheus
```

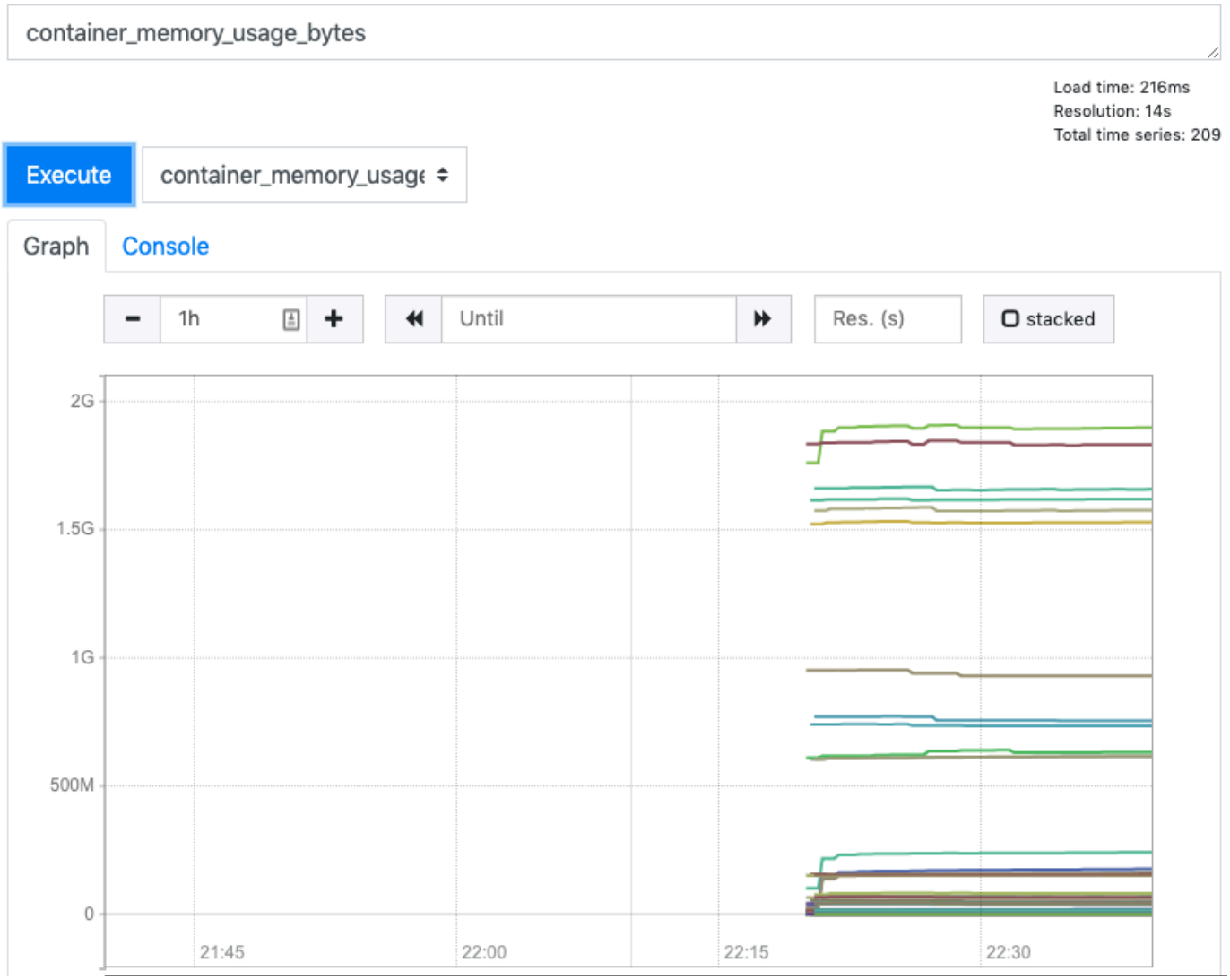
Veja um exemplo de saída abaixo.

```
NAME                                             READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-59b4c8c744-r7bgp         1/2     Running   0          48s
prometheus-kube-state-metrics-7cfd87cf99-jkz2f   1/1     Running   0          48s
prometheus-node-exporter-jcjqz                   1/1     Running   0          48s
prometheus-node-exporter-jxv2h                   1/1     Running   0          48s
prometheus-node-exporter-vbdks                   1/1     Running   0          48s
prometheus-pushgateway-76c444b68c-82tnw          1/1     Running   0          48s
prometheus-server-775957f748-mmht9               1/2     Running   0          48s
```

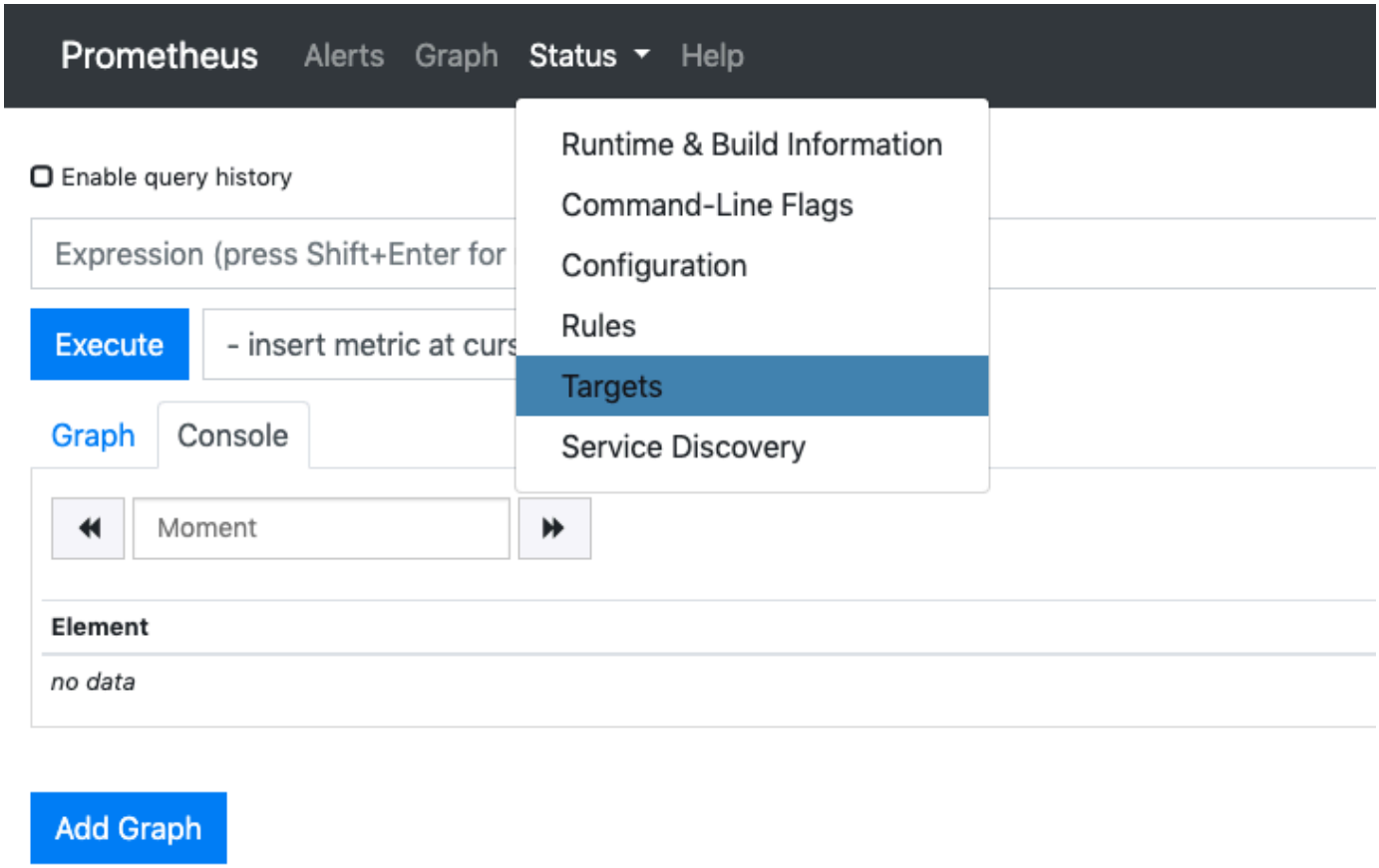
5. Use o `kubectl` para redirecionamento do console do Prometheus para sua máquina local.

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. Insira `http://localhost:9090` em um navegador da Web para visualizar o console do Prometheus.
7. Escolha uma métrica do menu - insert metric at cursor (- inserir métrica no cursor) e selecione Execute (Executar). Selecione a guia Graph (Gráfico) para mostrar a métrica ao longo do tempo. A imagem a seguir mostra `container_memory_usage_bytes` ao longo do tempo.



8. Na barra de navegação superior, selecione Status e Targets (Destinos).



Todos os endpoints do Kubernetes que estão conectados ao Prometheus usando detecção de serviço são exibidos.

Visualizar as métricas brutas do ambiente de gerenciamento no formato Prometheus

Como alternativa à implantação do Prometheus, o servidor da API do Kubernetes expõe várias métricas que são representadas em um formato do Prometheus. Essas métricas são úteis para monitoramento e análise. Elas são expostas internamente por meio de um endpoint de métricas que se refere à API HTTP `/metrics`. Assim como outros endpoints, esse endpoint é exposto no plano de controle do Amazon EKS. Esse endpoint é útil principalmente para analisar uma métrica específica. Para analisar métricas ao longo do tempo, recomendamos implantar o Prometheus.

Para visualizar a saída de métricas brutas, use `kubectl` com o sinalizador `--raw`. Esse comando permite passar qualquer caminho HTTP e retorna a resposta bruta.

```
kubectl get --raw /metrics
```

Veja um exemplo de saída abaixo.

```
[...]
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
 method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

Essa saída bruta retorna textualmente o que o servidor de API expõe. As diferentes métricas são listadas por linha, com cada linha incluindo um nome de métrica, tags e um valor.

```
metric_name{"tag"="value"[,...]}
            value
```

Monitorar dados de cluster com o Amazon CloudWatch

O Amazon CloudWatch Observability coleta logs, métricas e dados de rastreamento em tempo real. Ele os envia para o Amazon CloudWatch e o AWS X-Ray. É possível instalar este complemento para habilitar o CloudWatch Application Signals e o CloudWatch Container Insights com observabilidade aprimorada para o Amazon EKS. Isso ajuda você a monitorar a integridade e o desempenho da infraestrutura e de aplicações conteinerizadas. O Amazon CloudWatch Observability Operator foi desenvolvido para instalar e configurar os componentes necessários.

O Amazon EKS oferece suporte ao Amazon CloudWatch Observability Operator como um complemento do Amazon EKS. O complemento permite o Container Insights em ambos os nós

de processamento Linux e Windows no cluster. Para ativar o Container Insights no Windows, a versão do complemento do Amazon EKS deve ser a `1.5.0` ou superior. No momento, o CloudWatch Application Signals não é compatível com o Amazon EKS Windows.

Os tópicos abaixo descrevem como começar a utilizar o Amazon CloudWatch Observability Operator para o seu cluster do Amazon EKS.

- Para obter instruções sobre como instalar esse complemento, consulte Instalar o CloudWatch Agent usando os CloudWatch Observability do Amazon EKS no Guia do usuário do Amazon CloudWatch.
- Para obter mais informações sobre o CloudWatch Application Signals, consulte Application Signals.
- Para obter mais informações sobre o Container Insights, consulte Usar o Container Insights no Guia do usuário do Amazon CloudWatch.

Enviar logs do ambiente de gerenciamento para o CloudWatch Logs

O registro em log do plano de controle do Amazon EKS fornece logs de auditoria e diagnóstico diretamente do plano de controle do Amazon EKS para o CloudWatch Logs em sua conta. Esses logs facilitam a proteção e a execução dos clusters. Você pode selecionar os tipos exatos de logs que precisa, e eles serão enviados como fluxos de log para cada cluster do Amazon EKS no CloudWatch. Você pode usar filtros de assinatura do CloudWatch para fazer análises em tempo real nos logs ou encaminhá-los para outros serviços (os logs serão codificados em Base64 e compactados com o formato gzip). Para obter mais informações, consulte Registro em log do Amazon CloudWatch.

Você pode começar a usar o registro em log do plano de controle do Amazon EKS escolhendo os tipos de logs que deseja habilitar para cada cluster novo ou existente do Amazon EKS. É possível habilitar ou desabilitar cada tipo de log por cluster usando o AWS Management Console, a AWS CLI (versão `1.16.139` ou superior) ou por meio da API do Amazon EKS. Quando habilitados, os logs são enviados automaticamente do cluster do Amazon EKS para o CloudWatch Logs na mesma conta.

Ao usar o registro em log do plano de controle do Amazon EKS, você será cobrado pela definição de preço padrão do Amazon EKS para cada cluster executado. Você será cobrado pela ingestão de dados padrão do CloudWatch Logs e pelos custos de armazenamento de qualquer log enviado ao

CloudWatch Logs pelos clusters. Você também será cobrado por qualquer recurso da AWS, como instâncias do Amazon EC2 ou volumes do Amazon EBS, que forem provisionados como parte do cluster.

Os seguintes tipos de log do plano de controle do cluster estão disponíveis. Cada tipo de log corresponde a um componente do ambiente de gerenciamento do Kubernetes. Para saber mais sobre esses componentes, consulte Kubernetes Components (Componentes do Kubernets) na documentação do Kubernetes.

Servidor de API (**`api`**)

O servidor de API do cluster é o componente do ambiente de gerenciamento que expõe a API do Kubernetes. Se você ativar os logs do servidor de API ao iniciar o cluster, ou logo depois, os logs incluirão os sinalizadores do servidor de API que foram usados para iniciar o servidor de API. Para obter mais informações, consulte `kube-apiserver` e a audit policy (política de auditoria), na documentação do Kubernetes.

Auditoria (**`audit`**)

Os logs de auditoria do Kubernetes fornecem um registro dos usuários, administradores ou componentes individuais do sistema que afetaram o cluster. Para obter mais informações, consulte Auditing (Auditoria) na documentação do Kubernetes.

Autenticador (**`authenticator`**)

Os logs do autenticador são exclusivos do Amazon EKS. Esses logs representam o componente do ambiente de gerenciamento que o Amazon EKS usa para a autenticação de Controle de acesso com base em função (RBAC) do Kubernetes usando credenciais do IAM. Para obter mais informações, consulte Organizar e monitorar recursos de clusters.

Gerenciador do controlador (**`controllerManager`**)

O gerenciador do controlador gerencia os principais loops de controle que são fornecidos com o Kubernetes. Para obter mais informações, consulte kube-controller-manager na documentação do Kubernetes.

Agendador (**`scheduler`**)

O componente agendador gerencia quando e onde os Pods são executados no cluster. Para obter mais informações, consulte kube-scheduler na documentação do Kubernetes.

Habilitar ou desabilitar os logs do ambiente de gerenciamento

Por padrão, os logs do plano de controle de cluster não são enviados ao CloudWatch Logs. É necessário habilitar cada tipo de log individualmente para enviar logs ao cluster. As taxas de ingestão, armazenamento de arquivamento e verificação de dados do CloudWatch Logs se aplicam aos logs do plano de controle habilitados. Para obter mais informações, consulte Preço do CloudWatch.

Para atualizar a configuração de registro em log do ambiente de gerenciamento, o Amazon EKS requer até cinco endereços IP disponíveis em cada sub-rede. Quando você habilita um tipo de log, os logs são enviados com um nível de verbosidade de log de 2.

AWS Management Console

Para habilitar ou desabilitar os logs do plano de controle com a AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do cluster para exibir as informações dele.
3. Escolha a guia Observabilidade.
4. Na seção Log do ambiente de gerenciamento, escolha Gerenciar registro em log.
5. Para cada tipo de log individual, escolha se o tipo de log deve ser ativado ou desativado. Por padrão, o cada tipo de log está desativado.
6. Escolha Save changes (Salvar alterações) para terminar.

AWS CLI

Para habilitar ou desabilitar os logs do plano de controle com a AWS CLI

1. Verifique a versão do AWS CLI com o comando a seguir.

```
aws --version
```

Se sua versão da AWS CLI for anterior à versão `1.16.139`, primeiramente será necessário atualizar para a versão mais recente. Para instalar ou atualizar a AWS CLI, consulte Installing the AWS Command Line Interface (Instalar a CLI) no Manual do usuário do AWS Command Line Interface.

2. Atualize a configuração de exportação do log do plano de controle do cluster com o seguinte comando da AWS CLI. Substitua *my-cluster* pelo nome do cluster e especifique os valores desejados para acesso ao endpoint.

> **Note**
>
> O comando a seguir envia todos os tipos de log disponíveis ao CloudWatch Logs.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

Veja um exemplo de saída abaixo.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

3. Monitore o status da atualização da configuração do log com o comando a seguir, usando o nome do cluster e o ID da atualização que foram retornados pelo comando anterior. Sua atualização estará concluída quando o status aparecer como `Successful`.

```
aws eks describe-update \
    --region region-code\
    --name my-cluster \
```

```
    --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9
```

Veja um exemplo de saída abaixo.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "Successful",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

Visualizar os logs do ambiente de gerenciamento do cluster

Depois que tiver habilitado qualquer um dos tipos de log do plano de controle para o cluster do Amazon EKS, você poderá visualizá-los no console do CloudWatch.

Para saber mais sobre como visualizar, analisar e gerenciar logs no CloudWatch, consulte o Manual do usuário do Amazon CloudWatch Logs.

Para visualizar os logs do plano de controle do cluster no console do CloudWatch

1. Abra o console do CloudWatch. Esse link abre o console, exibe os grupos de log disponíveis atualmente e os filtra com o prefixo `/aws/eks`.
2. Escolha o cluster para o qual deseja visualizar os logs. O formato do nome do grupo de logs é `/aws/eks/my-cluster/cluster`.
3. Escolha o fluxo de logs que deseja visualizar. A lista a seguir descreve o formato do nome do fluxo de logs para cada tipo de log.

> **Note**
>
> À medida que os dados do fluxo de logs aumentam, os nomes dos fluxos de log são alternados. Quando existem vários fluxos de logs para um determinado tipo de log, você pode visualizar o fluxo de logs mais recente procurando o nome do fluxo de logs com a hora do último evento mais recente.

- KubernetesLogs do componente servidor de API (**`api`**): `kube-apiserver-`*`1234567890abcdef01234567890abcde`*
- Auditoria (**`audit`**) – `kube-apiserver-audit-`*`1234567890abcdef01234567890abcde`*
- Autenticador (**`authenticator`**) – `authenticator-`*`1234567890abcdef01234567890abcde`*
- Controller manager (**`controllerManager`**) (Gerenciador do controlador) – `kube-controller-manager-`*`1234567890abcdef01234567890abcde`*
- Scheduler (**`scheduler`**) – `kube-scheduler-`*`1234567890abcdef01234567890abcde`* (Agendador)

4. Revise os eventos do fluxo de logs.

   Por exemplo, você deve ver os sinalizadores iniciais do servidor de API para o cluster ao visualizar a parte superior de `kube-apiserver-`*`1234567890abcdef01234567890abcde`*.

   > **Note**
   >
   > Caso não veja os logs do servidor de API no começo do fluxo de log, provavelmente o arquivo de log do servidor de API foi alternado no servidor antes de você ativar o registro em log do servidor de API no servidor. Qualquer arquivo de log que for alternado antes que o registro em log do servidor de API seja ativado não poderá ser exportado para o CloudWatch.
   > Porém, é possível criar outro cluster com a mesma versão do Kubernetes e habilitar o registro em log do servidor de API ao criar o cluster. Os clusters com a mesma versão de plataforma têm os mesmos sinalizadores ativados, portanto, os sinalizadores devem corresponder aos sinalizadores do novo cluster. Ao terminar de visualizar os sinalizadores do novo cluster no CloudWatch, você poderá excluir o novo cluster.

Registrar chamadas de API em log como eventos do AWS CloudTrail

O Amazon ECR é integrado ao AWS CloudTrail. O CloudTrail é um serviço que fornece um registro das ações realizadas por um usuário, um perfil ou um serviço da AWS no Amazon EKS. O CloudTrail captura todas as chamadas de API para o Amazon EKS como eventos. Isso inclui chamadas do console do Amazon EKS e as chamadas de código para as operações de API do Amazon EKS.

Se você criar uma trilha, poderá habilitar a entrega contínua de eventos do CloudTrail para um bucket do Amazon S3. Isso inclui eventos para o Amazon EKS. Se você não configurar uma trilha, ainda poderá visualizar os eventos mais recentes no console do CloudTrail em Event history (Histórico de eventos). Utilizando as informações coletadas pelo CloudTrail, é possível determinar vários detalhes sobre uma solicitação. Por exemplo, é possível determinar quando a solicitação foi feita ao Amazon EKS, o endereço IP de onde ela foi feita e quem a fez.

Para saber mais sobre o CloudTrail, consulte o Guia do usuário do AWS CloudTrail.

Tópicos

Visualizar referências úteis para AWS CloudTrail

Quando você cria sua conta da AWS, o CloudTrail também está habilitado na conta da AWS. Quando ocorre qualquer atividade no Amazon EKS, essa atividade é registrada em um evento do CloudTrail junto com outros eventos de serviços da AWS no Event history (Histórico de eventos). Você pode visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Viewing events with CloudTrail event history.

Para obter um registro de eventos em andamento na sua conta da AWS, incluindo eventos do Amazon EKS, crie uma trilha. Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as Regiões da AWS. A trilha registra eventos no log de todas as Regiões da AWS na partição da AWS e entrega os arquivos de log ao bucket do Amazon S3 especificado por você. Além disso, é possível configurar outros serviços da AWS para analisar mais ainda mais e agir com base nos dados de

eventos coletados nos logs do CloudTrail. Para obter mais informações, consulte os recursos a seguir.

- Visão geral da criação de uma trilha
- Serviços e integrações compatíveis com o CloudTrail
- Configurar notificações do Amazon SNS para o CloudTrail
- Receber arquivos de log do CloudTrail de várias regiões e Receber arquivos de log do CloudTrail de várias contas

Todas as ações do Amazon EKS são registradas pelo CloudTrail e documentadas na Referência da API do Amazon EKS. Por exemplo, as chamadas para as seções `CreateCluster`, `ListClusters` e `DeleteCluster` geram entradas nos arquivos de log do CloudTrail.

Cada evento ou entrada de log contém informações sobre o tipo de identidade do IAM que fez a solicitação e quais credenciais foram usadas. Se forem usadas credenciais temporárias, a entrada mostrará como as credenciais foram obtidas.

Para mais informações, consulte Elemento userIdentity do CloudTrail.

Analisar entradas do arquivo de log do AWS CloudTrail

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log a um bucket do Amazon S3 especificado. Os arquivos de log do CloudTrail contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer origem e inclui informações sobre a ação solicitada. Isso inclui informações como a data e a hora da ação, bem como os parâmetros de solicitação que foram utilizados. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API pública, portanto não são exibidos em uma ordem específica.

O exemplo a seguir mostra uma entrada de log do CloudTrail que demonstra a ação `CreateCluster`.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/username",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
```

```
    "userName": "username"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "region-code",
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/6.4.0",
  "requestParameters": {
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-a670c2df",
        "subnet-4f8c5004"
      ]
    },
    "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
    "clusterName": "test"
  },
  "responseElements": {
    "cluster": {
      "clusterName": "test",
      "status": "CREATING",
      "createdAt": 1527535003.208,
      "certificateAuthority": {},
      "arn": "arn:aws:eks:region-code:111122223333:cluster/test",
      "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
      "version": "1.10",
      "resourcesVpcConfig": {
        "securityGroupIds": [],
        "vpcId": "vpc-21277358",
        "subnetIds": [
          "subnet-a670c2df",
          "subnet-4f8c5004"
        ]
      }
    }
  },
  "requestID": "a7a0735d-62ab-11e8-9f79-81ce5b2b7d37",
  "eventID": "eab22523-174a-499c-9dd6-91e7be3ff8e3",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
```

```
}
```

Entradas de log para funções vinculadas ao serviço do Amazon EKS

As funções vinculadas ao serviço do Amazon EKS fazem chamadas de API para recursos da AWS. Entradas de log do CloudTrail com `username: AWSServiceRoleForAmazonEKS` e `username: AWSServiceRoleForAmazonEKSNodegroup` aparecerão para chamadas feitas pelas funções vinculadas ao serviço do Amazon EKS. Para obter mais informações sobre o Amazon EKS e as funções vinculadas ao serviço, consulte Usar funções vinculadas ao serviço para o Amazon EKS.

O exemplo a seguir mostra uma entrada de log do CloudTrail que demonstra uma ação do `DeleteInstanceProfile` feita pela função vinculada ao serviço do `AWSServiceRoleForAmazonEKSNodegroup`, anotada no `sessionContext`.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3WHGPEZ7SJ2CW55C5:EKS",
        "arn": "arn:aws:sts::111122223333:assumed-role/
AWSServiceRoleForAmazonEKSNodegroup/EKS",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA3WHGPEZ7SJ2CW55C5",
                "arn": "arn:aws:iam::111122223333:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAmazonEKSNodegroup"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-02-26T00:56:33Z"
            }
        },
        "invokedBy": "eks-nodegroup.amazonaws.com"
    },
    "eventTime": "2020-02-26T00:56:34Z",
    "eventSource": "iam.amazonaws.com",
```

```
    "eventName": "DeleteInstanceProfile",
    "awsRegion": "region-code",
    "sourceIPAddress": "eks-nodegroup.amazonaws.com",
    "userAgent": "eks-nodegroup.amazonaws.com",
    "requestParameters": {
        "instanceProfileName": "eks-11111111-2222-3333-4444-abcdef123456"
    },
    "responseElements": null,
    "requestID": "11111111-2222-3333-4444-abcdef123456",
    "eventID": "11111111-2222-3333-4444-abcdef123456",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Visualizar métricas para grupos do Amazon EC2 Auto Scaling

Os grupos de nós gerenciados pelo Amazon EKS têm as métricas de grupo do Amazon EC2 Auto Scaling habilitadas por padrão, sem custo adicional. O grupo do Auto Scaling envia dados de amostragem ao Amazon CloudWatch a cada minuto. Essas métricas podem ser refinadas pelo nome dos grupos do Auto Scaling. Eles fornecem visibilidade contínua do histórico do grupo do Auto Scaling que alimenta seus grupos de nós gerenciados, como alterações no tamanho do grupo ao longo do tempo. As métricas do grupo do Auto Scaling estão disponíveis no console do Auto Scaling ou no console do Amazon CloudWatch. Para obter mais informações, consulte Monitorar métricas do CloudWatch para seus grupos e instâncias do Auto Scaling.

Com a coleta de métricas de grupos do Auto Scaling, você é capaz de monitorar a escalação de grupos de nós gerenciados. As métricas do grupo do Auto Scaling informam o tamanho mínimo, máximo e desejado de um grupo do Auto Scaling. Você pode criar um alarme se o número de nós em um grupo de nós ficar abaixo do tamanho mínimo, o que indicaria um grupo de nós não íntegro. O rastreamento do tamanho do grupo de nós também é útil para ajustar o número máximo para a capacidade do plano de dados não se esgote.

Se você preferir que essas métricas não sejam coletadas, poderá optar por desabilitar todas ou apenas algumas delas. Por exemplo, você pode fazer isso para evitar ruídos nos seus painéis do CloudWatch. Para obter mais informações, consulte Métricas de uso do Amazon CloudWatch para Amazon EC2 Auto Scaling.

Enviar dados de métricas e rastreamento com o ADOT Operator

O Amazon EKS é compatível com o uso do AWS Management Console, da AWS CLI e da API do Amazon EKS para instalar e gerenciar o Operador AWS Distro for OpenTelemetry (ADOT). Isso facilita habilitar suas aplicações executadas no Amazon EKS para enviar dados de métricas e de rastreamento a várias opções de serviço de monitoramento, como o Amazon CloudWatch, o Prometheus e o X-Ray.

Para obter informações adicionais, consulte Conceitos básicos do AWS Distro para OpenTelemetry usando complementos do EKS na documentação do AWS Distro para OpenTelemetry.

Mais serviços da AWS integrados ao Amazon EKS

Além dos serviços abordados em outras seções, o Amazon EKS trabalha com mais serviços da AWS para fornecer soluções adicionais. Este tópico identifica alguns dos outros serviços que usam o Amazon EKS para adicionar funcionalidades ou serviços que o Amazon EKS usa para executar tarefas.

Tópicos

Criar recursos do Amazon EKS com o AWS CloudFormation

O Amazon EKS é integrado ao AWS CloudFormation, um serviço que ajuda você a modelar e configurar seus recursos da AWS, para que você possa passar menos tempo criando e gerenciando seus recursos e sua infraestrutura. Crie um modelo que descreva todos os recursos da AWS que você quiser, por exemplo, um cluster do Amazon EKS, e a AWS CloudFormation cuidará do provisionamento e da configuração desses recursos para você.

Quando você usa o AWS CloudFormation, é possível reutilizar seu modelo para configurar os recursos do Amazon EKS repetidamente e de forma consistente. Basta descrever seus recursos uma vez e, depois, provisionar os mesmos recursos repetidamente em várias contas e regiões da AWS.

Amazon EKS e modelos do AWS CloudFormation

Para provisionar e configurar recursos para o Amazon EKS e serviços relacionados, você deve entender os modelos do AWS CloudFormation. Os modelos são arquivos de texto formatados em JSON ou YAML. Esses modelos descrevem os recursos que você deseja provisionar nas

suas pilhas do AWS CloudFormation. Se você não estiver familiarizado com JSON ou YAML, poderá usar o AWS CloudFormation Designer para ajudá-lo a começar a usar os modelos do AWS CloudFormation. Para obter mais informações, consulte O que é o Designer? (O que é o AWS CloudFormation Designer) no Manual do usuário do AWS CloudFormation.

O Amazon EKS oferece suporte à criação de clusters e grupos de nós no AWS CloudFormation. Para obter mais informações, incluindo exemplos de modelos JSON e YAML para os recursos do Amazon EKS, consulte Amazon EKS resource type reference (Referência do tipo de recurso do Amazon EKS) no Manual do usuário do AWS CloudFormation.

Saiba mais sobre o AWS CloudFormation

Para saber mais sobre o AWS CloudFormation, consulte os seguintes recursos:

- AWS CloudFormation
- Manual do usuário do AWS CloudFormation
- Guia do usuário da interface de linha de comando do AWS CloudFormation

Amazon EKS e zonas locais da AWS

Uma zona local da AWS é uma extensão de uma Região da AWS na proximidade geográfica de seus usuários. O Local Zones tem suas próprias conexões com a Internet e suporte no AWS Direct Connect. Os recursos criados em uma zona local podem atender usuários locais com comunicações de latência baixa. Para obter mais informações, consulte Local Zones.

O Amazon EKS é compatível com determinados recursos em zonas locais. Isso inclui nós do Amazon EC2 autogerenciados, volumes do Amazon EBS e Application Load Balancers (ALBs). Recomendamos que você considere o seguinte quando usar zonas locais como parte do cluster do Amazon EKS:

Nós

Você não pode criar grupos de nós gerenciados ou nós do Fargate em Zonas locais com o Amazon EKS. Porém, é possível criar nós do Amazon EC2 autogerenciados em zonas locais utilizando a API do Amazon EC2, AWS CloudFormation ou `eksctl`. Para obter mais informações, consulte Fazer a manutenção dos nós por conta própria com nós autogerenciados.

Arquitetura de rede

- O ambiente de gerenciamento do Kubernetes gerenciado pelo Amazon EKS é sempre executado em Região da AWS. O ambiente de gerenciamento do Kubernetes gerenciado pelo Amazon EKS não pode ser executado em Local Zone. Como Local Zones aparecem como uma sub-rede na VPC, o Kubernetes considera os recursos de Local Zone como parte dessa sub-rede.
- O cluster do Amazon EKS Kubernetes se comunica com as instâncias do Amazon EC2 que você executa na Região da AWS ou zona local e usando as interfaces de rede elástica gerenciadas pelo Amazon EKS. Para saber mais sobre a arquitetura de rede do Amazon EKS, consulte Rede Amazon EKS.
- Ao contrário das sub-redes regionais, o Amazon EKS não pode colocar interfaces de rede nas sub-redes da zona local. Isso significa que você não deve especificar sub-redes da zona local ao criar o cluster.

Deep Learning Containers

Os Deep Learning Containers da AWS são um conjunto de imagens do Docker para treinar e fornecer modelos em TensorFlow no Amazon EKS e no Amazon Elastic Container Service (Amazon ECS). Deep Learning Containers fornecem ambientes otimizados com bibliotecas TensorFlow, NVIDIA CUDA (para instâncias de GPU) e Intel MKL (para instâncias de CPU) e estão disponíveis no Amazon ECR.

Para começar a usar os AWS Deep Learning Containers no Amazon EKS, consulte Configuração do Amazon EKS no Guia do desenvolvedor do AWS Deep Learning Containers.

Amazon VPC Lattice

O Amazon VPC Lattice é um serviço de rede de aplicações totalmente gerenciado, criado diretamente na infraestrutura de rede da AWS, que você pode usar para conectar, proteger e monitorar seus serviços em várias contas e nuvens privadas virtuais (VPCs). Com o Amazon EKS, é possível utilizar o Amazon VPC Lattice usando o AWS Gateway API Controller, uma implementação da Kubernetes Gateway API. Com o Amazon VPC Lattice, você configura a conectividade entre clusters com a semântica padrão do Kubernetes de maneira simples e consistente. Para começar a usar o Amazon VPC Lattice com o Amazon EKS, consulte o Guia do usuário do AWS Gateway API Controller.

AWS Resilience Hub

O AWS Resilience Hub avalia a resiliência de um cluster do Amazon EKS analisando sua infraestrutura. O AWS Resilience Hub usa a configuração de controle de acesso com base em funções (RBAC) do Kubernetes para avaliar as workloados Kubernetes implantadas no seu cluster. Para obter mais informações, consulte Habilitar o acesso ao seu cluster do Amazon EKSAWS Resilience Hub, no Guia do usuário do AWS Resilience Hub.

Detectar ameaças com o Amazon GuardDuty

O Amazon GuardDuty é um serviço de detecção de ameaças que ajuda a proteger contas, contêineres, workloads e dados no ambiente da AWS. Usando modelos de machine learning (ML) e recursos de detecção de anomalias e ameaças, o GuardDuty monitora continuamente diferentes fontes de log e atividades de runtime para identificar e priorizar possíveis riscos de segurança e atividades maliciosas no seu ambiente.

Entre outros recursos, o GuardDuty oferece os dois recursos a seguir que detectam possíveis ameaças aos seus clusters do EKS: Proteção do EKS e Monitoramento de runtime.

Proteção do EKS

> Esse recurso fornece cobertura de detecção de ameaças para ajudar você a proteger clusters do Amazon EKS monitorando os logs de auditoria do Kubernetes associados. Os logs de auditoria do Kubernetescapturam ações sequenciais em seu cluster, incluindo atividades de usuários e aplicações usando a API e o ambiente de gerenciamento do Kubernetes. Por exemplo, o GuardDuty pode identificar que APIs chamadas para potencialmente adulterar recursos em um cluster do Kubernetes foram invocadas por um usuário não autenticado.
>
> Quando a Proteção do EKS estiver habilitada, o GuardDuty poderá acessar seus logs de auditoria do Amazon EKS somente para detecção contínua de ameaças. Se o GuardDuty identificar uma possível ameaça ao seu cluster, ele gerará uma descoberta de log de auditoria do Kubernetes associado de um tipo específico. Para obter mais informações sobre os tipos de descobertas disponíveis nos logs de auditoria do Kubernetes, consulte os tipos de descoberta dos logs de auditoria do Kubernetes no Guia do usuário do Amazon GuardDuty.
>
> Para obter mais informações, consulte Proteção do EKS no Guia do usuário do Amazon GuardDuty.

Monitoramento de runtime

Esse recurso monitora e analisa eventos em nível de sistema operacional, rede e arquivos para ajudar você a detectar possíveis ameaças em workloads do AWS específicas do seu ambiente.

Quando você habilita o Monitoramento de runtime e instala o agente do GuardDuty em seus clusters do Amazon EKS, o GuardDuty começa a monitorar os eventos de runtime associados a esse cluster. Se o GuardDuty identificar uma possível ameaça ao seu cluster, ele gerará uma descoberta de Monitoramento de runtime. Por exemplo, uma ameaça pode começar comprometendo um único contêiner que executa um aplicativo Web vulnerável. Esse aplicativo Web pode ter permissões de acesso aos contêineres e workloads subjacentes. Nesse cenário, credenciais configuradas incorretamente podem levar a um acesso mais amplo à conta e aos dados nela armazenados.

Para configurar o Monitoramento de runtime, instale o agente do GuardDuty no seu cluster como um complemento do Amazon EKS. Para obter mais informações sobre o complemento, consulte Complementos do Amazon EKS disponíveis da AWS.

Para obter mais informações, consulte Monitoramento de runtime no Guia do usuário do Amazon GuardDuty.

Como usar o Amazon Security Lake com o Amazon EKS

O Amazon Security Lake é um serviço de data lake de segurança totalmente gerenciado que permite centralizar dados de segurança de várias fontes, incluindo o Amazon EKS. Ao integrar o Amazon EKS com o Security Lake, você pode obter insights mais profundos sobre as atividades realizadas em seus recursos do Kubernetes e aprimorar a postura de segurança de seus clusters do Amazon EKS.

> **Note**
>
> Para obter mais informações sobre como usar o Security Lake com o Amazon EKS e configurar fontes de dados, consulte a documentação do Amazon Security Lake.

Como usar o Security Lake com o Amazon EKS

Dados de segurança centralizados: o Security Lake coleta e centraliza automaticamente dados de segurança de seus clusters do Amazon EKS, juntamente com dados de outros serviços da AWS,

provedores de SaaS, fontes on-premises e fontes de terceiros. Isso fornece uma visão abrangente de sua postura de segurança em toda a sua organização.

Formato de dados padronizado: o Security Lake converte os dados coletados para o formato Open Cybersecurity Schema Framework (OCSF), que é um esquema padrão de código aberto. Essa normalização permite uma análise mais fácil e a integração com outras ferramentas e serviços de segurança.

Detecção aprimorada de ameaças: ao analisar os dados de segurança centralizados, incluindo os registros do ambiente de gerenciamento do Amazon EKS, você pode detectar atividades potencialmente suspeitas em seus clusters do Amazon EKS com mais eficiência. Isso ajuda a identificar e responder prontamente aos incidentes de segurança.

Gerenciamento de dados simplificado: o Security Lake gerencia o ciclo de vida de seus dados de segurança com configurações personalizáveis de retenção e replicação. Isso simplifica as tarefas de gerenciamento de dados e garante que você retenha os dados necessários para fins de conformidade e auditoria.

Habilitar o Security Lake para Amazon EKS

Para começar a usar o Security Lake com o Amazon EKS, siga estas etapa:

1. Habilite o ambiente de gerenciamento do Amazon EKS para seus clusters do EKS. Consulte Habilitar e desabilitar os logs do ambiente de gerenciamento para obter instruções detalhadas.
2. Adicione os logs de auditoria do Amazon EKS como fonte no Security Lake. Em seguida, o Security Lake começará a coletar informações detalhadas sobre as atividades realizadas nos recursos do Kubernetes em execução em seus clusters do EKS.
3. Configure as opções de retenção e replicação para seus dados de segurança no Security Lake com base em seus requisitos.
4. Use os dados de OCSF normalizados armazenados no Security Lake para resposta a incidentes, análise de segurança e integração com outros serviços da AWS ou ferramentas de terceiros. Por exemplo, você pode Gerar insights de segurança a partir dos dados do Amazon Security Lake usando o Amazon OpenSearch Ingestion.

Analisar logs do EKS no Security Lake

O Security Lake normaliza os eventos de log do EKS para o formato OCSF, facilitando a análise e a correlação dos dados com outros eventos de segurança. Você pode usar várias ferramentas e

serviços, como Amazon Athena, Amazon QuickSight ou ferramentas de análise de segurança de terceiros, para consultar e visualizar os dados normalizados.

Para obter mais informações sobre o mapeamento OCSF para eventos de log do EKS, consulte a referência de mapeamento no repositório OCSF no GitHub.

Amazon Detective

O Amazon Detective ajuda a analisar, investigar e identificar rapidamente a causa raiz de descobertas de segurança ou atividades suspeitas. O Detective coleta automaticamente dados de log dos seus recursos da AWS. Em seguida, ele usa machine learning, análises estatísticas e a teoria de grafos para gerar visualizações que ajudam a realizar investigações de segurança eficazes com maior rapidez. O Detective faz a pré-construção de agregações de dados, resumos e contexto predefinidos que podem ajudar você a analisar e determinar a natureza e a extensão de possíveis problemas de segurança. Para obter mais informações, consulte o Guia do usuário do Amazon Detective.

O Detective organiza dados Kubernetes e AWS em descobertas como:

- Detalhes do cluster do Amazon EKS, incluindo a identidade IAM que criou o cluster e o perfil de serviço do cluster. Você pode investigar a atividade de API AWS e Kubernetes dessas identidades do IAM com o Detective.
- Detalhes do contêiner, como a imagem e o contexto de segurança. Também é possível revisar detalhes de encerrado Pods.
- A atividade de API Kubernetes, incluindo tendências gerais na atividade de API e detalhes sobre chamadas de API específicas. Por exemplo, você pode mostrar o número de chamadas de API Kubernetes bem-sucedidas e malsucedidas que foram emitidas durante um intervalo de tempo selecionado. Além disso, a seção sobre chamadas de API recém-observadas pode ser útil para identificar atividades suspeitas.

Logs de auditoria do Amazon EKS são um pacote de origem de dados opcional que pode ser adicionado ao seu gráfico de comportamento do Detective. Você pode visualizar os pacotes de origem opcionais disponíveis e seu status na sua conta. Para obter mais informações, consulte Logs de auditoria do Amazon EKS para Detective e o Guia do usuário do Amazon Detective.

Usar o Amazon Detective com o Amazon EKS

Para revisar descobertas de um cluster do Amazon EKS

Antes de poder revisar as descobertas, o Detective deve estar habilitado por pelo menos 48 horas na mesma Região da AWS em que seu cluster está localizado. Para obter mais informações, consulte Configurar o Amazon Detective no Guia do usuário do Amazon Detective.

1. Abra o console do Detective em https://console.aws.amazon.com/detective/.
2. No painel de navegação à esquerda, selecione Search (Pesquisar).
3. Selecione Choose type (Escolher tipo) e, em seguida, selecione EKS cluster.
4. Insira o nome do cluster ou ARN e escolha Search (Pesquisar).
5. Nos resultados da pesquisa, escolha o nome do cluster cujas atividades deseja visualizar. Para obter mais informações sobre o que você pode visualizar, consulte Atividade geral da API Kubernetes envolvendo um cluster do Amazon EKS no Guia do usuário do Amazon Detective.

Solucionar problemas com clusters e nós do Amazon EKS

Este capítulo aborda alguns erros comuns que você poderá encontrar ao usar o Amazon EKS e como resolvê-los. Se você precisar solucionar problemas em áreas específicas do Amazon EKS, consulte os tópicos Solução de problemas do IAM, Solução de problemas do Amazon EKS Connector e Solução de problemas do ADOT utilizando complementos do EKS.

Para obter outras informações sobre solução de problemas, consulte o conteúdo do Centro de conhecimento sobre o Amazon Elastic Kubernetes Service em AWS re:Post.

Insufficient capacity (Capacidade insuficiente)

Se você receber o seguinte erro ao tentar criar um cluster do Amazon EKS, isso significa que uma das zonas de disponibilidade que você especificou não tem capacidade suficiente para oferecer suporte a um cluster.

```
Cannot create cluster 'example-cluster' because region-1d, the targeted Availability Zone, does not currently have sufficient capacity to support the cluster. Retry and choose from these Availability Zones: region-1a, region-1b, region-1c
```

Tente criar o cluster com sub-redes na VPC do cluster que são hospedadas nas zonas de disponibilidade retornadas por essa mensagem de erro.

Existem zonas de disponibilidade nas quais um cluster não pode residir. Compare as zonas de disponibilidade em que suas sub-redes estão com a lista de zonas de disponibilidade no Requisitos e considerações para sub-redes.

Worker nodes fail to join cluster (Falha nos nós ao ingressar no cluster)

Há alguns motivos comuns que impedem que os nós ingressem no cluster:

- Se os nós forem nós gerenciados, o Amazon EKS adiciona entradas ao `ConfigMap` `aws-auth` quando você cria o grupo de nós. Se a entrada tiver sido removida ou modificada, você precisará adicioná-la novamente. Para obter mais informações, insira **`eksctl create`**

`iamidentitymapping --help` no seu terminal. É possível verificar a versão atual das entradas do `ConfigMap` `aws-auth` ao substituir *`my-cluster`* no comando a seguir pelo nome do seu cluster e, depois, executar o comando modificado: **`eksctl get iamidentitymapping --cluster`** ***`my-cluster`***. O ARN da função que você especifica não pode incluir um caminho diferente de `/`. Por exemplo, se o nome da sua função for `development/apps/my-role`, você precisará alterá-lo para `my-role` ao especificar o ARN da função. Verifique se o ARN do perfil do IAM do nó (não o ARN do perfil de instância) está especificado.

Se os nós forem autogerenciados e você não tiver criado entradas de acesso para o ARN do perfil do IAM do nó, execute os mesmos comandos listados para os nós gerenciados. Se você criou uma entrada de acesso para o ARN do perfil do IAM do nó, talvez ela não esteja configurada corretamente na entrada de acesso. Verifique se o ARN do perfil do IAM do nó (não o ARN do perfil de instância) está especificado como o ARN principal na entrada `ConfigMap` `aws-auth` ou na entrada de acesso. Para obter mais informações sobre as entradas de acesso, consulte Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS.

- O ClusterName no modelo do AWS CloudFormation do seu nó não corresponde exatamente ao nome do cluster no qual você deseja que seus nós ingressem. O fornecimento de um valor incorreto a esse campo resulta em uma configuração incorreta do arquivo `/var/lib/kubelet/kubeconfig` do nó, e os nós não ingressarão no cluster.
- O nó não é marcado como sendo de propriedade do cluster. Os nós devem ter a seguinte etiqueta aplicada a eles, onde *`my-cluster`* é substituída pelo nome do cluster.

Chave	Valor
`kubernetes.io/cluster/` *`my-cluster`*	owned

- Talvez não seja possível que os nós acessem o cluster usando um endereço IP público. Verifique se os nós implantados em sub-redes públicas recebem um endereço IP público. Caso contrário, é possível associar um endereço IP elástico a um nó depois que ele for executado. Para obter mais informações, consulte Associating an Elastic IP address with a running instance or network interface (Associar um endereço IP elástico a uma instância em execução ou interface de rede). Se a sub-rede pública não estiver definida para atribuir automaticamente endereços IP públicos a instâncias implantadas nela, recomendamos ativar essa configuração. Para obter mais informações, consulte Modificar o atributo de endereçamento IPv4 público para a sua sub-rede. Se o nó for implantado em uma sub-rede privada, a sub-rede deverá ter uma rota para um gateway NAT que tenha um endereço IP público atribuído a ele.

- O endpoint do AWS STS para a Região da AWS na qual você está implantando os nós não está habilitado para sua conta. Para habilitar a região, consulte Ativação e desativação do AWS STS em uma Região da AWS.
- O nó não tem uma entrada DNS privada, resultando no erro `node "" not found` no log do `kubelet`. Verifique se a VPC na qual o nó foi criado tem valores definidos para `domain-name` e `domain-name-servers` como `Options` em um `DHCP options set`. Os valores padrão são `domain-name:<region>.compute.internal` e `domain-name-servers:AmazonProvidedDNS`. Para obter mais informações, consulte Conjuntos de opções DHCP no Guia do usuário da Amazon VPC.
- Se os nós do grupo de nós gerenciados não se conectarem ao cluster em 15 minutos, um problema de integridade “NoDecreationFailure” será emitido e o status do console será definido como `Create failed`. Para AMIs do Windows com tempos de lançamento lentos, esse problema pode ser resolvido usando o lançamento rápido.

Você pode usar o runbook `AWSSupport-TroubleshootEKSWorkerNode` para identificar e solucionar problemas de causas comuns que impedem o ingresso dos nós de processamento em um cluster. Para obter mais informações, consulte `AWSSupport-TroubleshootEKSWorkerNode` no Referência de runbook do AWS Systems Manager Automation.

Acesso negado ou não autorizado (**`kubectl`**)

Se você receber um dos erros a seguir ao executar os comandos do `kubectl`, é porque o `kubectl` não está configurado corretamente para o Amazon EKS ou as credenciais para a entidade principal do IAM (perfil ou usuário) que estão sendo usadas não estão mapeadas para um nome de usuário do Kubernetes com permissões suficientes para objetos do Kubernetes no seu cluster do Amazon EKS.

- `could not get token: AccessDenied: Access denied`
- `error: You must be logged in to the server (Unauthorized)`
- `error: the server doesn't have a resource type "svc"`

Isso pode ocorrer devido a um dos seguintes motivos:

- O cluster foi criado com credenciais para uma entidade principal do IAM e o `kubectl` está configurado para usar credenciais para uma entidade principal do IAM diferente. Para resolver essa situação, atualize seu arquivo `kube config` para usar as credenciais que criaram o cluster.

Para obter mais informações, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig.

- Se o seu cluster atender aos requisitos mínimos de plataforma na seção de pré-requisitos de Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS, não existirá uma entrada de acesso com a entidade principal do IAM. Se ela existir, ela não terá os nomes de grupo do Kubernetes necessários definidos para ela ou não terá a política de acesso adequada associada a ela. Para obter mais informações, consulte Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS.
- Se seu cluster não atender aos requisitos mínimos da plataforma em Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS, não existirá uma entrada com a entidade principal do IAM no `ConfigMap` `aws-auth`. Se existir, ela não será mapeada para nomes de grupos do Kubernetes vinculados a `Role` ou `ClusterRole` do Kubernetes com as permissões necessárias. Para obter mais informações sobre os objetos de autorização baseada em função (RBAC) do Kubernetes, consulte Using RBAC authorization na documentação do Kubernetes. É possível verificar a versão atual das entradas do `ConfigMap` `aws-auth` ao substituir *`my-cluster`* no comando a seguir pelo nome do seu cluster e, depois, executar o comando modificado: **`eksctl get iamidentitymapping --cluster`** ***`my-cluster`***. Se uma entrada com o ARN da entidade principal do IAM não estiver no `ConfigMap`, insira **`eksctl create iamidentitymapping --help`** no seu terminal para saber como criar uma.

Se você instalar e configurar a AWS CLI, poderá configurar as credenciais do IAM usadas. Para obter mais informações, consulte Configuração da AWS CLI no Guia do usuário da AWS Command Line Interface. Também é possível configurar `kubectl` para usar um perfil do IAM, se você assumir um perfil do IAM para acessar objetos do Kubernetes no seu cluster. Para obter mais informações, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig.

`hostname doesn't match`

Seu sistema precisa ter a versão `2.7.9` ou superior do Python. Caso contrário, você receberá erros de `hostname doesn't match` com chamadas do AWS CLI para o Amazon EKS. Para obter mais informações, consulte What are "hostname doesn't match" errors? nas Perguntas frequentes sobre solicitações do Python.

`getsockopt: no route to host`

O Docker é executado no intervalo `172.17.0.0/16` do CIDR em clusters do Amazon EKS. Recomendamos que as sub-redes da VPC do cluster não sobreponham esse intervalo. Caso contrário, você receberá o seguinte erro:

```
Error: : error upgrading connection: error dialing backend: dial tcp
 172.17.<nn>.<nn>:10250: getsockopt: no route to host
```

`Instances failed to join the Kubernetes cluster`

Se você receber o erro `Instances failed to join the Kubernetes cluster` no AWS Management Console, certifique-se de que o acesso ao endpoint privado do cluster esteja habilitado ou que você configurou corretamente os blocos CIDR para acesso ao endpoint público. Para obter mais informações, consulte Controlar o acesso à rede ao endpoint do servidor de API do cluster.

Códigos de erro para o grupo de nós gerenciados

Se o grupo de nós gerenciados encontrar um problema de integridade de hardware, o Amazon EKS retornará um código de erro para ajudar você a diagnosticar o problema. Essas verificações de integridade não detectam problemas de software porque são baseadas em verificações de integridade do Amazon EC2. A lista a seguir descreve os códigos de erro.

AccessDenied

> O Amazon EKS ou um ou mais nós gerenciados não conseguem autenticar nem autorizar com o servidor de API do cluster do Kubernetes. Para obter mais informações sobre como resolver uma causa comum, consulte Como corrigir uma causa comum de erros `AccessDenied` para grupos de nós gerenciados. As AMIs privadas do Windows também podem causar esse código de erro juntamente com a mensagem de erro `Not authorized for images`. Para obter mais informações, consulte `Not authorized for images`.

AmiIdNotFound

> Não foi possível encontrar o ID da AMI associada ao seu modelo de inicialização. Certifique-se de que a AMI existe e é compartilhada com sua conta.

AutoScalingGroupNotFound

Não foi possível encontrar o grupo do Auto Scaling associado ao grupo de nós gerenciados. Você pode recriar um grupo do Auto Scaling com as mesmas configurações para fazer a recuperação.

ClusterUnreachable

O Amazon EKS ou um ou mais nós gerenciados não conseguem se comunicar com o servidor de API do cluster do Kubernetes. Isso pode acontecer se houver interrupções na rede ou se os servidores de API estiverem expandindo solicitações de processamento.

Ec2SecurityGroupNotFound

Não foi possível localizar o grupo de segurança do cluster. Você deve recriar seu cluster.

Ec2SecurityGroupDeletionFailure

Não foi possível excluir o grupo de segurança de acesso remoto para o grupo de nós gerenciados. Remova quaisquer dependências do grupo de segurança.

Ec2LaunchTemplateNotFound

Não foi possível encontrar o modelo de inicialização do Amazon EC2 para o grupo de nós gerenciados. Você deve recriar seu grupo de nós para a recuperação.

Ec2LaunchTemplateVersionMismatch

A versão do modelo de inicialização do Amazon EC2 para o grupo de nós gerenciados não corresponde à versão que o Amazon EKS criou. Você pode reverter para a versão criada pelo Amazon EKS para fazer a recuperação.

IamInstanceProfileNotFound

Não foi possível encontrar o perfil de instância do IAM para o grupo de nós gerenciados. Você pode recriar um perfil de instância com as mesmas configurações para fazer a recuperação.

IamNodeRoleNotFound

Não foi possível encontrar o perfil do IAM para o grupo de nós gerenciados. Você pode recriar uma função do IAM com as mesmas configurações para fazer a recuperação.

AsgInstanceLaunchFailures

O grupo do Auto Scaling está sofrendo falhas ao tentar iniciar instâncias.

NodeCreationFailure

As instâncias iniciadas não conseguem se registrar no cluster do Amazon EKS. Causas comuns dessa falha são permissões insuficientes da função do IAM do nó ou a falta de acesso à Internet na saída para os nós. Seus nós devem atender a qualquer um dos seguintes requisitos:

- Capacidade de acessar a Internet utilizando um endereço IP público. O grupo de segurança associado à sub-rede em que o nó se encontra deve permitir a comunicação. Para ter mais informações, consulte Requisitos e considerações para sub-redes e Considerações e requisitos sobre grupos de segurança do Amazon EKS.
- Os nós e a VPC devem atender aos requisitos em Implementar clusters privados com acesso limitado à internet.

InstanceLimitExceeded

Sua conta da AWS não pode iniciar mais nenhuma instância do tipo especificado. Você pode solicitar um aumento do limite de instâncias do Amazon EC2 para fazer a recuperação.

InsufficientFreeAddresses

Uma ou mais sub-redes associadas ao grupo de nós gerenciados não têm endereços IP disponíveis suficientes para novos nós.

InternalFailure

Esses erros geralmente são causados por um problema no servidor do Amazon EKS.

Como corrigir uma causa comum de erros **`AccessDenied`** para grupos de nós gerenciados

A causa mais comum de erros `AccessDenied` ao executar operações em grupos de nós gerenciados, é a ausência de `eks:node-manager`, `ClusterRole` ou `ClusterRoleBinding`. O Amazon EKS configura esses recursos no cluster como parte da integração com grupos de nós gerenciados, e eles são necessários para gerenciar os grupos de nós.

O `ClusterRole` pode mudar ao longo do tempo, mas deve ser semelhante ao seguinte exemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
```

```
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
```

O `ClusterRoleBinding` pode mudar ao longo do tempo, mas deve ser semelhante ao seguinte exemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Verifique se o `eks:node-manager` `ClusterRole` existe.

```
kubectl describe clusterrole eks:node-manager
```

Se estiver presente, compare a saída com ao exemplo de `ClusterRole` anterior.

Verifique se o `eks:node-manager` `ClusterRoleBinding` existe.

```
kubectl describe clusterrolebinding eks:node-manager
```

Se estiver presente, compare a saída com ao exemplo de `ClusterRoleBinding` anterior.

Se você identificou a ausência de `ClusterRole` ou `ClusterRoleBinding` como a causa de um erro `AcessDenied` ao solicitar operações de grupo de nó gerenciado, você pode restaurá-los. Salve o conteúdo a seguir em um arquivo denominado *`eks-node-manager-role.yaml`*.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
```

```
  - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Aplique o arquivo.

```
kubectl apply -f eks-node-manager-role.yaml
```

Tente novamente a operação do grupo de nós para ver se isso resolveu o problema.

`Not authorized for images`

Uma possível causa de uma mensagem de erro `Not authorized for images` é usar uma AMI privada do Windows do Amazon EKS para executar grupos de nós gerenciados do Windows. Depois de lançar novas AMIs do Windows, a AWS torna privadas as AMIs com mais de quatro meses, o que as torna inacessíveis. Se o seu grupo de nós gerenciados estiver usando uma AMI do Windows privada, considere atualizar o grupo de nós gerenciados do Windows. Embora não possamos garantir que poderemos fornecer acesso a AMIs que se tornaram privadas, você pode solicitar acesso ao preencher um tíquete no AWS Support. Para obter mais informações, consulte Patches, atualizações de segurança e IDs de AMI no Guia do usuário do Amazon EC2.

O nó está no estado **`NotReady`**

Se o seu nó entrar em um status de `NotReady`, isso provavelmente indica que ele não está íntegro e está indisponível para a programação de novos Pods. Isso pode ocorrer por vários motivos, como a falta de recursos suficientes para a CPU, a memória ou o espaço em disco disponível no nó.

Para AMIs do Windows otimizadas para o Amazon EKS, não há reserva para recursos de computação especificados por padrão na configuração `kubelet`. Para ajudar a evitar problemas de

recursos, é possível reservar recursos de computação para processos do sistema, fornecendo ao `kubelet` valores de configuração para `kube-reserved` ou `system-reserved`. Você faz isso ao usar o parâmetro de linha de comando `-KubeletExtraArgs` no script de inicialização. Para obter mais informações, consulte Reserve Compute Resources for System Daemons na documentação do Kubernetes e Bootstrap script configuration parameters neste guia do usuário.

Ferramenta de coleta de logs da CNI

O Amazon VPC CNI plugin for para Kubernetes tem seu próprio script de solução de problemas que está disponível em nós em `/opt/cni/bin/aws-cni-support.sh`. Você pode usar o script para coletar logs de diagnóstico para casos de suporte e solução de problemas gerais.

Use o seguinte comando para executar o script em seu nó:

```
sudo bash /opt/cni/bin/aws-cni-support.sh
```

> **Note**
>
> Se o script não está presente no local, ocorreu uma falha na execução do contêiner do CNI. Você pode baixar e executar o script manualmente com o seguinte comando:
>
> ```
> curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/log-collector-script/linux/eks-log-collector.sh
> sudo bash eks-log-collector.sh
> ```

O script coleta as seguintes informações de diagnóstico. A versão da CNI que você implantou pode ser anterior à versão do script.

```
      This is version 0.6.1. New versions can be found at https://github.com/awslabs/
amazon-eks-ami

Trying to collect common operating system logs...
Trying to collect kernel logs...
Trying to collect mount points and volume information...
Trying to collect SELinux status...
Trying to collect iptables information...
Trying to collect installed packages...
```

```
Trying to collect active system services...
Trying to collect Docker daemon information...
Trying to collect kubelet information...
Trying to collect L-IPAMD information...
Trying to collect sysctls information...
Trying to collect networking information...
Trying to collect CNI configuration information...
Trying to collect running Docker containers and gather container data...
Trying to collect Docker daemon logs...
Trying to archive gathered information...

 Done... your bundled logs are located in /var/
log/eks_i-0717c9d54b6cfaa19_2020-03-24_0103-UTC_0.6.1.tar.gz
```

As informações de diagnóstico são coletadas e armazenadas em:

```
/var/log/eks_i-0717c9d54b6cfaa19_2020-03-24_0103-UTC_0.6.1.tar.gz
```

A rede de runtime do contêiner não está pronta

Você pode receber um erro `Container runtime network not ready` e erros de autorização semelhantes a:

```
4191 kubelet.go:2130] Container runtime network not ready: NetworkReady=false
 reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config
 uninitialized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
4191 kubelet_node_status.go:106] Unable to register node
 "ip-10-40-175-122.ec2.internal" with API server: Unauthorized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
```

Isso pode acontecer devido a um dos seguintes motivos:

1. Ou você não tem um `ConfigMap` `aws-auth` em seu cluster ou ele não inclui entradas para o perfil do IAM com o qual você configurou seus nós.

   A entrada `ConfigMap` será necessária se os nós atenderem a um dos seguintes critérios:

   - Nós gerenciados em um cluster com qualquer versão do Kubernetes ou da plataforma.

- Nós autogerenciados em um cluster anterior a uma das versões da plataforma listadas na seção de pré-requisitos do tópico Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS.

Para resolver o problema, verifique as entradas existentes no `ConfigMap` ao substituir *my-cluster* no comando a seguir pelo nome do cluster e execute o comando modificado: **eksctl get iamidentitymapping --cluster *my-cluster***. Se você receber uma mensagem de erro do comando, talvez seja porque seu cluster não tem um `ConfigMap` `aws-auth`. O comando a seguir adiciona uma entrada ao `ConfigMap`. Se o `ConfigMap` não existir, ele também será criado pelo comando. Substitua *111122223333* pelo ID da Conta da AWS do perfil do IAM e *myAmazonEKSNodeRole* pelo nome da função do seu nó.

```
eksctl create iamidentitymapping --cluster my-cluster \
    --arn arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --group
 system:bootstrappers,system:nodes \
    --username system:node:{{EC2PrivateDNSName}}
```

O ARN da função que você especifica não pode incluir um caminho diferente de `/`. Por exemplo, se o nome da sua função for `development/apps/my-role`, você precisará alterá-lo para `my-role` ao especificar o ARN da função. Verifique se o ARN do perfil do IAM do nó (não o ARN do perfil de instância) está especificado.

2. Seus nós autogerenciados estão em um cluster com uma versão da plataforma na versão mínima listada nos pré-requisitos do tópico Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS, mas uma entrada não está listada no `ConfigMap` `aws-auth` (consulte o item anterior) para o perfil do IAM do nó ou não existe uma entrada de acesso para a função. Para resolver o problema, visualize as entradas de acesso existentes ao substituir *my-cluster* no comando a seguir pelo nome do cluster e execute o comando modificado: **aws eks list-access-entries --cluster-name *my-cluster***. O comando a seguir adiciona uma entrada de acesso para o perfil do IAM do nó. Substitua *111122223333* pelo ID da Conta da AWS do perfil do IAM e *myAmazonEKSNodeRole* pelo nome da função do seu nó. Se você tiver um nó do Windows, substitua *EC2_Linux* por **EC2_Windows**. Verifique se o ARN do perfil do IAM do nó (não o ARN do perfil de instância) está especificado.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --type EC2_Linux
```

Tempo limite de handshake TLS

Quando um nó não conseguir estabelecer uma conexão com o endpoint do servidor de API público, talvez você veja um erro semelhante ao erro a seguir.

```
server.go:233] failed to run Kubelet: could not init cloud provider "aws": error
 finding instance i-1111f2222f333e44c: "error listing AWS instances: \"RequestError:
 send request failed\\ncaused by: Post  net/http: TLS handshake timeout\""
```

O processo `kubelet` gerará e testará continuamente o endpoint do servidor de API. O erro também pode ocorrer temporariamente durante algum procedimento que realize uma atualização contínua do cluster no plano de controle, como uma alteração de configuração ou atualização de versão.

Para resolver o problema, verifique a tabela de rotas e os grupos de segurança para garantir que o tráfego dos nós possa atingir o endpoint público.

InvalidClientTokenId

Se estiver usando perfis do IAM para contas de serviço para um Pod ou `DaemonSet` implantado em um cluster em uma Região da AWS da China e ainda não tiver definido a variável de ambiente `AWS_DEFAULT_REGION` na especificação, o Pod ou `DaemonSet` poderá receber o seguinte erro:

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation:
 The security token included in the request is invalid
```

Para resolver o problema, é necessário adicionar a variável de ambiente `AWS_DEFAULT_REGION` à especificação do Pod ou `DaemonSet`, conforme exibido na seguinte especificação do Pod de exemplo:

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
```

```
    env:
    - name: AWS_DEFAULT_REGION
      value: "region-code"
```

Expiração do certificado webhook de admissão da VPC

Se o certificado usado para assinar o webhook de admissão da VPC expirar, o status das novas implantações do Pod do Windows permanecerá em `ContainerCreating`.

Para resolver o problema, se você tiver suporte ao Windows herdado em plano de dados, consulte Renovar o certificado webhook de admissão da VPC. Se a versão do cluster e da plataforma forem posteriores às versões listadas em Pré-requisitos de suporte do Windows, recomendamos remover o suporte ao Windows herdado do plano de dados e habilitá-lo no ambiente de gerenciamento. Ao fazer isso, você não precisará gerenciar o certificado webhook. Para obter mais informações, consulte Implantar nós do Windows em clusters do EKS.

Os grupos de nós devem corresponder à versão do Kubernetes antes da atualização do ambiente de gerenciamento

Antes de atualizar um ambiente de gerenciamento para uma nova versão do Kubernetes, a versão secundária dos nós gerenciados e do Fargate no cluster deve ser a mesma da versão atual do ambiente de gerenciamento. A API `update-cluster-version` do Amazon EKS rejeita solicitações até que você atualize todos os nós gerenciados do Amazon EKS para a versão atual do cluster. O Amazon EKS fornece APIs para atualizar nós gerenciados. Para obter informações sobre como atualizar a versão do Kubernetes do grupo de nós gerenciado, consulte Atualizar um grupo de nós gerenciados para seu cluster. Para atualizar a versão de um nó do Fargate, exclua o pod representado pelo nó e reimplante o pod depois de atualizar o ambiente de gerenciamento. Para obter mais informações, consulte Atualizar um cluster existente para a nova versão do Kubernetes.

Ao iniciar muitos nós, ocorrem erros **`Too Many Requests`**

Se você iniciar muitos nós ao mesmo tempo, poderá ver uma mensagem de erro nos logs de execução de Dados do usuário do Amazon EC2 que indica `Too Many Requests`. Isso pode ocorrer porque o ambiente de gerenciamento está sendo sobrecarregado por chamadas `describeCluster`. A sobrecarga resulta em controle de utilização, nós com falha ao executar o script de bootstrap e nós que não são completamente incorporados ao cluster.

Verifique se os argumentos `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` estão sendo transmitidos ao script de bootstrap do nó. Ao incluir esses argumentos, não há necessidade de o script bootstrap criar uma chamada `describeCluster`, o que ajuda a evitar a sobrecarga do ambiente de gerenciamento. Para obter mais informações, consulte Fornecer dados de usuário para passar argumentos para o arquivo `bootstrap.sh` incluído com uma AMI do Linux/Bottlerocket otimizada para o Amazon EKS.

Resposta de erro HTTP 401 unauthorized (HTTP 401 não autorizado) em solicitações do servidor de API do Kubernetes

Esses erros serão mostrados se um token de conta de serviço de um Pod tiver expirado em um cluster.

O servidor de APIs Kubernetes do seu cluster do Amazon EKS rejeita solicitações com tokens com mais de 90 dias. Em versões anteriores do Kubernetes, os tokens não tinham expiração. Isso significa que os clientes que dependem desses tokens precisam atualizá-los em até uma hora. Para evitar que o servidor de API do Kubernetes rejeite a solicitação devido a um token inválido, a versão do SDK de cliente do Kubernetes utilizada pela workload deve ser igual ou posterior às seguintes:

- Versão do Go `0.15.7` e posteriores
- Versão do Python `12.0.0` e posteriores
- Versão Java `9.0.0` e posterior
- Versão JavaScript `0.10.3` e posterior
- Ramificação Ruby `master`
- Versão do Haskell `0.3.0.0`
- Versão do C# `7.0.5` ou posterior.

É possível identificar todos os Pods existentes no cluster que estão usando tokens obsoletos. Para obter mais informações, consulte Contas de serviço do Kubernetes.

A versão da plataforma Amazon EKS está mais de duas versões atrás da versão atual da plataforma

Isso pode acontecer quando o Amazon EKS não consegue atualizar automaticamente versão da plataforma do cluster. Embora existem muitas causas para isso, veja a seguir algumas das causas

comuns. Se algum desses problemas se aplicar ao cluster, ele ainda poderá funcionar; apenas a versão da plataforma não será atualizada pelo Amazon EKS.

Problema

O perfil do IAM do cluster foi excluído; essa função foi especificada quando o cluster foi criado. Você pode verificar qual função foi especificada com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut
 -d / -f 2
```

Veja um exemplo de saída abaixo.

```
eksClusterRole
```

Solução

Crie um novo perfil do IAM do cluster com o mesmo nome.

Problema

Uma sub-rede especificada durante a criação do cluster foi excluída; as sub-redes a serem usadas com o cluster foram especificadas durante a criação do cluster. Você pode verificar quais sub-redes foram especificadas com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

Veja um exemplo de saída abaixo.

```
[
"subnet-EXAMPLE1",
"subnet-EXAMPLE2"
]
```

Solução

Confirme se os IDs de sub-rede existem na conta.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
```

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --query
 "Subnets[*].SubnetId"
```

Veja um exemplo de saída abaixo.

```
[
"subnet-EXAMPLE3",
"subnet-EXAMPLE4"
]
```

Se os IDs de sub-rede retornados na saída não corresponderem aos IDs de sub-rede especificados quando o cluster foi criado, se você quiser que o Amazon EKS atualize o cluster, será necessário alterar as sub-redes usadas pelo cluster. Isso ocorre porque, se você especificou mais de duas sub-redes ao criar o cluster, o Amazon EKS selecionará aleatoriamente as sub-redes que você especificou para a criação de novas interfaces. Essas interfaces de rede permitem que o ambiente de gerenciamento se comunique com os nós. O Amazon EKS não atualizará o cluster se a sub-rede selecionada não existir. Você não tem controle sobre qual das sub-redes especificadas na criação do cluster o Amazon EKS escolhe para criar uma nova interface de rede.

Quando você inicia uma atualização de versão do Kubernetes para o cluster, a atualização pode falhar pelo mesmo motivo.

Problema

Um grupo de segurança especificado durante a criação do cluster foi excluído. Se você especificou grupo de segurança durante a criação do cluster, poderá ver os IDs com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.securityGroupIds
```

Veja um exemplo de saída abaixo.

```
[
    "sg-EXAMPLE1"
]
```

Se `[]` for retornado, nenhum grupo de segurança foi especificado quando o cluster foi criado e a falta de um grupo de segurança não é o problema. Se grupos de segurança forem retornados, confirme se eles existem na conta.

Solução

Confirme se esses grupos de segurança existem na conta.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$vpc_id" --query
 "SecurityGroups[*].GroupId"
```

Veja um exemplo de saída abaixo.

```
[
"sg-EXAMPLE2"
]
```

Se os IDs dos grupos de segurança retornados na saída não corresponderem aos IDs dos grupos de segurança especificados quando o cluster foi criado, se você quiser que o Amazon EKS atualize o cluster, será necessário alterar os grupos de segurança usados pelo cluster. O Amazon EKS não atualizará um cluster se os IDs de grupo de segurança especificados na criação do cluster não existirem.

Quando você inicia uma atualização de versão do Kubernetes para o cluster, a atualização pode falhar pelo mesmo motivo.

Outros motivos pelos quais o Amazon EKS não atualiza a versão da plataforma do cluster

- Você não tem pelo menos seis (embora recomendemos 16) endereços IP disponíveis em cada uma das sub-redes especificadas ao criar o cluster. Se não tiver endereços IP disponíveis suficientes na sub-rede, você precisará liberar endereços IP na sub-rede ou alterar as sub-redes usadas pelo cluster para usar sub-redes com endereços IP disponíveis suficientes.
- Você habilitou a criptografia de segredos quando você criou o cluster e a chave do AWS KMSque você especificou foi excluída. Se você quiser que o Amazon EKS atualize o cluster, precisará criar um novo cluster

Perguntas frequentes sobre a integridade do cluster e códigos de erro com caminhos de resolução

O Amazon EKS detecta problemas com os clusters do EKS e com a infraestrutura do cluster e os armazena na integridade do cluster. Você pode detectar, solucionar problemas e resolver problemas

de cluster mais rapidamente com a ajuda das informações de integridade do cluster. Isso permite que você crie ambientes de aplicativos mais seguros e atualizados. Além disso, pode ser impossível atualizar para versões mais recentes do Kubernetes ou instalar atualizações de segurança do Amazon EKS em um cluster degradado como resultado de problemas com a infraestrutura ou a configuração do cluster necessárias. O Amazon EKS pode levar 3 horas para detectar problemas ou detectar que um problema foi resolvido.

A integridade de um cluster do Amazon EKS é uma responsabilidade compartilhada entre o Amazon EKS e seus usuários. Você é responsável pela infraestrutura de pré-requisitos dos perfis do IAM e das sub-redes do Amazon VPC, bem como por outras infraestruturas necessárias, que devem ser fornecidas com antecedência. O Amazon EKS detecta mudanças na configuração dessa infraestrutura e do cluster.

Para acessar a integridade do cluster no console do Amazon EKS, procure uma seção chamada Problemas de integridade na guia Visão geral da página de detalhes do cluster do Amazon EKS. Esses dados também estarão disponíveis ao chamar a ação `DescribeCluster` na API do EKS, por exemplo, de dentro do AWS Command Line Interface.

Por que devo usar esse recurso?

> Você terá maior visibilidade da integridade do cluster do Amazon EKS, diagnosticará e corrigirá rapidamente quaisquer problemas, sem precisar perder tempo depurando ou abrindo casos de suporte da AWS. Por exemplo: você excluiu acidentalmente uma sub-rede do cluster do Amazon EKS; o Amazon EKS não poderá criar interfaces de rede entre contas e os comandos da AWS CLI do Kubernetes, como exec do `kubectl` ou logs do`kubectl`. Eles falharão com o erro: `Error from server: error dialing backend: remote error: tls: internal error.` Agora você verá um problema de integridade do Amazon EKS que diz: `subnet-da60e280 was deleted: could not create network interface`.

Como este recurso se relaciona ou funciona com outros serviços da AWS?

> Os perfis do IAM e as sub-redes do Amazon VPC são dois exemplos de infraestrutura de pré-requisito com a qual a integridade do cluster detecta problemas. O recurso retornará informações detalhadas se esses recursos não estiverem configurados corretamente.

Um cluster com problemas de integridade gera custos?

> Sim, cada cluster do Amazon EKS é cobrado de acordo com o preço padrão do Amazon EKS. O recurso de integridade do cluster está disponível sem custo adicional.

Esse recurso funciona com clusters do Amazon EKS no AWS Outposts?

Sim, problemas de cluster são detectados em clusters do EKS na Nuvem AWS, incluindo clusters estendidos no AWS Outposts e clusters locais no AWS Outposts. A integridade do cluster não detecta problemas com o Amazon EKS Anywhere ou o Amazon EKS Distro (EKS-D).

Posso ser notificado quando novos problemas forem detectados?

Não, você precisa verificar o console do Amazon EKS ou chamar a API `DescribeCluster` do EKS.

O console me avisa sobre problemas de integridade?

Sim, qualquer cluster com problemas de integridade incluirá um banner na parte superior do console.

As duas primeiras colunas são o que é necessário para os valores de resposta da API. O terceiro campo do objeto Health ClusterIssue é resourceIds, cujo retorno depende do tipo de problema.

Código	Message	ResourceIds	Cluster recuperável?	
SUBNET_NOT_FOUND	Não foi possível encontrar uma ou mais sub-redes que estão atualmente associadas ao seu cluster. Chame a API update-cluster-config do Amazon EKS para atualizar sub-redes.	IDs de sub-rede	Sim	
SECURITY_GROUP_NOT_FOUND	Não foi possível encontrar um ou mais grupos de segurança que estão atualmente associados ao seu cluster. Chame a API update-cluster-config do Amazon EKS para atualizar grupos de segurança	IDs de grupos de segurança	Sim	
IP_NOT_AVAILABLE	Uma ou mais sub-redes associadas ao seu cluster não têm	IDs de sub-rede	Sim	

Código	Message	ResourceIds	Cluster recuperável?	
	endereços IP disponíveis suficientes para o Amazon EKS realizar operações de gerenciamento de cluster. Libere endereços na(s) sub-rede(s) ou associe diferentes sub-redes ao seu cluster usando a API update-cluster-config do Amazon EKS.			
VPC_NOT_FOUND	Não foi possível encontrar a VPC associada ao seu cluster. Você deve excluir e recriar seu cluster.	ID da VPC	Não	
ASSUME_ROLE_ACCESS_DENIED	Seu cluster não está usando a função vinculada ao serviço do Amazon EKS. Não foi possível assumir a função associada ao seu cluster para realizar as operações de gerenciamento necessárias do Amazon EKS. Verifique se a função existe e tem a política de confiança necessária.	O perfil do IAM do cluster	Sim	

Código	Message	ResourceIds	Cluster recuperável?	
PERMISSION_ACCESS_DENIED	Seu cluster não está usando a função vinculada ao serviço do Amazon EKS. A função associada ao seu cluster não concede permissões suficientes para que o Amazon EKS execute as operações de gerenciamento necessárias. Verifique as políticas anexadas à função de cluster e se alguma política de negação separada foi aplicada.	O perfil do IAM do cluster	Sim	
ASSUME_ROLE_ACCESS_DENIED_USING_SLR	Não foi possível assumir a função vinculada ao serviço de gerenciamento de clusters do Amazon EKS. Verifique se a função existe e tem a política de confiança necessária.	A função vinculada ao serviço do Amazon EKS	Sim	
PERMISSION_ACCESS_DENIED_USING_SLR	A função vinculada ao serviço de gerenciamento de clusters do Amazon EKS não concede permissões suficientes para que o Amazon EKS execute as operações de gerenciamento necessárias. Verifique as políticas anexadas à função de cluster e se alguma política de negação separada foi aplicada.	A função vinculada ao serviço do Amazon EKS	Sim	

Código	Message	ResourceIds	Cluster recuperável?	
OPT_IN_REQUIRED	Sua conta não tem uma assinatura de serviço do Amazon EC2. Atualize as assinaturas da sua conta na página de configurações da conta.	N/D	Sim	
STS_REGIONAL_ENDPOINT_DISABLED	O endpoint regional do STS está desativado. Habilite o endpoint do Amazon EKS para realizar as operações de gerenciamento de cluster necessárias.	N/D	Sim	
KMS_KEY_DISABLED	A chave do AWS KMS associada ao seu cluster está desativada. Habilite novamente a chave para recuperar o cluster.	A KMS Key Arn	Sim	
KMS_KEY_NOT_FOUND	Não foi possível encontrar a chave do AWS KMS associada ao seu cluster. Você precisa excluir e recriar o cluster.	A KMS Key ARN	Não	
KMS_GRANT_REVOKED	As concessões para a chave do AWS KMS associada ao seu cluster foram revogadas. Você precisa excluir e recriar o cluster.	A KMS Key Arn	Não	

Conecte um cluster do Kubernetes a um console de gerenciamento do Amazon EKS com o Amazon EKS Connector

Você pode usar o Amazon EKS Connector para registrar e conectar qualquer cluster do Kubernetes à AWS compatível e visualizá-lo no console do Amazon EKS. Quando um cluster é conectado, você pode ver o status, a configuração e as workloads desse cluster no console do Amazon EKS. É possível usar esse recurso para visualizar clusters conectados no console do Amazon EKS, mas não é possível gerenciá-los. O Amazon EKS Connector também é um projeto de código aberto no Github. Para obter conteúdo técnico adicional, incluindo perguntas frequentes e solução de problemas, consulte Solução de problemas do Amazon EKS Connector

O Amazon EKS Connector pode conectar os tipos de clusters do Kubernetes a seguir ao Amazon EKS.

- Clusters do Kubernetes on-premises
- Clusters autogerenciados que estão em execução no Amazon EC2
- Clusters gerenciados de outros provedores de nuvem

Considerações sobre o Amazon EKS Connector

Antes de usar o Amazon EKS Connector, entenda:

- Você deve ter privilégios administrativos no cluster do Kubernetes para conectá-lo ao Amazon EKS.
- O cluster do Kubernetes deve ter nós de processamento do Linux de 64 bits (x86) presentes antes da conexão. Os nós de processamento do ARM não são compatíveis.
- Você deve ter nós de processamento no cluster do Kubernetes que tenham acesso de saída aos endpoints `ssm.` e `ssmmessages.` do Systems Manager. Para obter mais informações, consulte Endpoints do Systems Manager na Referência geral da AWS.
- Por padrão, é possível conectar até dez clusters em uma região. Você pode solicitar um aumento pelo console da cota de serviço. Consulte Solicitar um aumento de cota para obter mais informações.

- Somente as APIs `RegisterCluster`, `ListClusters`, `DescribeCluster` e `DeregisterCluster` do Amazon EKS são compatíveis com clusters externos do Kubernetes.
- É necessário ter as seguintes permissões para registrar um cluster:
  - eks:RegisterCluster
  - ssm:CreateActivation
  - ssm:DeleteActivation
  - iam:PassRole
- É necessário ter as seguintes permissões para cancelar o registro de um cluster:
  - eks:DeregisterCluster
  - ssm:DeleteActivation
  - ssm:DeregisterManagedInstance

Funções do IAM necessárias para o Amazon EKS Connector

O uso do Amazon EKS Connector requer estas duas funções do IAM:

- O perfil vinculado ao serviço do Amazon EKS Connector é criado quando você registra o cluster.
- Você deve criar o perfil do IAM de agente do Amazon EKS Connector. Para mais detalhes, consulte Função do IAM do conector do Amazon EKS.

Para habilitar a permissão de visualização de cluster e workload para as entidades principais do IAM, é necessário aplicar `eks-connector` e as funções de cluster do Amazon EKS Connector ao cluster. Siga as etapas em Conceder acesso para visualizar recursos de clusters do Kubernetes em um console do Amazon EKS.

Conectar um cluster externo do Kubernetes ao console de gerenciamento do Amazon EKS

Você pode conectar um cluster externo do Kubernetes ao Amazon EKS usando vários métodos no processo descrito a seguir. Esse processo envolve duas etapas: registrar o cluster no Amazon EKS e instalar o agente do `eks-connector` no cluster.

> **Important**
>
> Você deve concluir a segunda etapa em até 3 dias após a conclusão da primeira etapa, antes que o registro expire.

Métodos de conector

Nem todos os métodos de instalação do agente podem ser usados após cada um dos métodos de registro do cluster. A tabela a seguir lista cada método de registro e quais métodos de instalação do agente podem ser usados.

Etapa	Métodos		
Registrar os clusters	AWS Management Console	AWS Command Line Interface	`eksctl`
Instalar o agente do	Helm, manifestos YAML	Helm, manifestos YAML	Manifestos YAML

Pré-requisitos

- Certifique-se de que o perfil do agente do Amazon EKS Connector foi criado. Siga as etapas em Criar a função do IAM para o agente de conector do Amazon EKS.
- É necessário ter as seguintes permissões para registrar um cluster:
  - `eks:RegisterCluster`
  - `ssm:CreateActivation`
  - `ssm:DeleteActivation`
  - `iam:PassRole`

Etapa 1: Registrar o cluster

AWS CLI

Pré-requisitos

- A AWS CLI deve ser instalada. Para instalar ou atualizar, consulte Instalar a AWS CLI.

Para registrar o cluster com a AWS CLI

- Para a configuração do conector, especifique a função do IAM do agente do Amazon EKS Connector. Para obter mais informações, consulte Funções do IAM necessárias para o Amazon EKS Connector.

```
aws eks register-cluster \
     --name my-first-registered-cluster \
     --connector-config roleArn=arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole,provider="OTHER" \
     --region aws-region
```

Veja um exemplo de saída abaixo.

```
{
    "cluster": {
        "name": "my-first-registered-cluster",
        "arn": "arn:aws:eks:region:111122223333:cluster/my-first-registered-
cluster",
        "createdAt": 1627669203.531,
        "ConnectorConfig": {
            "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
            "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
            "activationExpiry": 1627672543.0,
            "provider": "OTHER",
            "roleArn": "arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole"
        },
        "status": "CREATING"
    }
}
```

Você usará os valores de `aws-region`, `activationId` e `activationCode` na próxima etapa.

AWS Management Console

Para registrar o cluster do Kubernetes no console.

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha Add cluster (Adicionar cluster) e selecione Register (Registrar) para abrir a página de configuração.
3. Na seção Configure cluster (Configurar cluster), preencha os seguintes campos:
   - Name (Nome) – um nome exclusivo para o cluster.
   - Provider (Fornecedor): escolha para exibir a lista suspensa de provedores de cluster do Kubernetes. Se não souber qual é o provedor específico, selecione Outro.
   - EKS Connector role (Função do EKS Connector): Selecione a função a ser usada para conectar o cluster.
4. Selecione Register cluster (Registrar cluster).
5. A página de visão geral do cluster é exibida. Se você quiser usar o chart do Helm, copie o comando `helm install` e prossiga na próxima etapa. Se desejar usar o manifesto YAML, escolha Baixar arquivo YAML para baixar o arquivo manifesto na unidade local.

> **Important**
>
> - Esta é sua única oportunidade de copiar o comando `helm install` ou baixar esse arquivo. Não saia dessa página, pois o link não estará acessível, e você precisará cancelar o registro do cluster e iniciar as etapas desde o início.
> - O comando ou o arquivo do manifesto só podem ser usados uma vez para o cluster registrado. Se você excluir recursos do cluster do Kubernetes, deve registrar novamente o cluster e obter um novo arquivo manifesto.

Prossiga para a próxima etapa para aplicar o arquivo manifesto ao cluster do Kubernetes.

eksctl

Pré-requisitos

- O `eksctl` versão `0.68` ou posterior deve estar instalado. Para obter informações sobre instalação ou upgrade, consulte Conceitos básicos do Amazon EKS: `eksctl`.

Para registrar o cluster no **`eksctl`**

1. Registre o cluster informando um nome, um provedor e uma região.

```
eksctl register cluster --name my-cluster --provider my-provider --
region region-code
```

Resultado do exemplo:

```
2021-08-19 13:47:26 [#]  creating IAM role "eksctl-20210819194112186040"
2021-08-19 13:47:26 [#]  registered cluster "<name>" successfully
2021-08-19 13:47:26 [#]  wrote file eks-connector.yaml to <current directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-clusterrole.yaml to <current
 directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-console-dashboard-full-access-
group.yaml to <current directory>
2021-08-19 13:47:26 [!]  note: "eks-connector-clusterrole.yaml" and "eks-
connector-console-dashboard-full-access-group.yaml" give full EKS Console access
 to IAM identity "<aws-arn>", edit if required; read https://eksctl.io/usage/
eks-connector for more info
2021-08-19 13:47:26 [#]  run `kubectl apply -f eks-connector.yaml,eks-connector-
clusterrole.yaml,eks-connector-console-dashboard-full-access-group.yaml` before
 expiry> to connect the cluster
```

Isso cria arquivos no seu computador local. Esses arquivos devem ser aplicados ao cluster externo em até três dias, ou o registro expira.

2. Em um terminal que possa acessar o cluster, aplique o arquivo `eks-connector-binding.yaml`:

```
kubectl apply -f eks-connector-binding.yaml
```

Etapa 2: instalar o agente do **`eks-connector`**

Helm chart

1. Se você usou a AWS CLI na etapa anterior, substitua o `ACTIVATION_CODE` e `ACTIVATION_ID` no comando a seguir pelos valores `activationId` e `activationCode` respectivamente. Substitua a `aws-region` pela Região da AWS que você usou na etapa anterior. Execute o seguinte comando para instalar o agente do `eks-connector` no cluster sendo resgistrado:

   ```
   $ helm install eks-connector \
     --namespace eks-connector \
     oci://public.ecr.aws/eks-connector/eks-connector-chart \
     --set eks.activationCode=ACTIVATION_CODE \
     --set eks.activationId=ACTIVATION_ID \
     --set eks.agentRegion=aws-region
   ```

   Se você usou o AWS Management Console na etapa anterior, use o comando que você copiou da etapa anterior que tem esses valores preenchidos.

2. Verifique a integridade da implantação do `eks-connector` instalado e aguarde que o status do cluster registrado no Amazon EKS seja `ACTIVE`.

YAML manifest

Faça a conexão aplicando o arquivo manifesto do Amazon EKS Connector ao cluster do Kubernetes. Para isso, você deve usar os métodos descritos anteriormente. Se o manifesto não for aplicado em até três dias, a validade do registro do Amazon EKS Connector expirará. Se a conexão do cluster expirar, o registro do cluster deve ser cancelado antes de ser conectado novamente.

1. Baixe o arquivo YAML do Amazon EKS Connector.

   ```
   curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/
   eks-connector/latest/eks-connector.yaml
   ```

2. Edite o arquivo YAML do Amazon EKS Connector para substituir todas as referências de `%AWS_REGION%`, `%EKS_ACTIVATION_ID%` e `%EKS_ACTIVATION_CODE%` por `aws-region`, `activationId` e `activationCode` da saída da etapa anterior.

O exemplo de comando a seguir pode substituir esses valores.

```
sed -i "s~%AWS_REGION%~$aws-region~g; s~%EKS_ACTIVATION_ID
%~$EKS_ACTIVATION_ID~g; s~%EKS_ACTIVATION_CODE%~$(echo -n $EKS_ACTIVATION_CODE |
 base64)~g" eks-connector.yaml
```

> **Important**
>
> Certifique-se de que seu código de ativação esteja no formato de base64.

3. Em um terminal que possa acessar o cluster, você pode aplicar o arquivo de manifesto atualizado, executando o seguinte comando:

```
kubectl apply -f eks-connector.yaml
```

4. Depois que os arquivos YAML de associação de função e manifesto do Amazon EKS Connector forem aplicados ao cluster do Kubernetes, confirme se o cluster está conectado.

```
aws eks describe-cluster \
     --name "my-first-registered-cluster" \
     --region AWS_REGION
```

   A saída deve incluir `status=ACTIVE`.

5. (Opcional) Adicione etiquetas ao seu cluster. Para obter mais informações, consulte Organizar recursos do Amazon EKS com tags.

Próximas etapas

Se você tiver algum problema com essas etapas, consulte Solução de problemas do Amazon EKS Connector.

Para conceder acesso às entidades principais do IAM ao console do Amazon EKS para visualizar recursos do Kubernetes em um cluster conectado, consulte Conceder acesso para visualizar recursos de clusters do Kubernetes em um console do Amazon EKS.

Conceder acesso para visualizar recursos de clusters do Kubernetes em um console do Amazon EKS

Conceda às entidades principais do IAM acesso ao console do Amazon EKS para visualizar informações sobre recursos do Kubernetes em execução no cluster conectado.

Pré-requisitos

A entidade principal do IAM que você usa para acessar o AWS Management Console deve atender aos seguintes requisitos:

- Você deve ter a permissão `eks:AccessKubernetesApi` do IAM.
- A conta do Amazon EKS Connector Service pode representar o IAM ou a entidade principal no cluster. Isso permite que o Amazon EKS Connector mapeie o usuário ou a entidade principal do IAM para um usuário do Kubernetes.

Para criar e aplicar a função de cluster do Amazon EKS Connector

1. Baixe o modelo de função do cluster do `eks-connector`.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-clusterrole.yaml
   ```

2. Edite o arquivo YAML do template de função de cluster. Substitua as referências de `%IAM_ARN%` pelo nome do recurso da Amazon (ARN) da entidade principal do IAM.
3. Aplique o YAML da função do cluster do Amazon EKS Connector ao cluster do Kubernetes.

   ```
   kubectl apply -f eks-connector-clusterrole.yaml
   ```

Para que uma entidade principal do IAM visualize recursos do Kubernetes no console do Amazon EKS, ela deve estar associada a um Kubernetes, `role` ou `clusterrole` com as permissões necessárias para ler os recursos. Para mais informações, consulte Using RBAC Authorization (Usar autorização RBAC) na documentação do Kubernetes.

Para configurar um entidade principal do IAM para acessar o cluster conectado

1. Você pode baixar um desses dois exemplos de arquivo de manifesto para criar um `clusterrole` e `clusterrolebinding` ou um `role` e `rolebinding`, respectivamente:

   Visualizar recursos do Kubernetes em todos os namespaces

   O perfil do cluster `eks-connector-console-dashboard-full-access-clusterrole` dá acesso a todos os namespaces e recursos que podem ser visualizados no console. Você pode alterar o nome da `role`, `clusterrole` e sua respectiva vinculação antes de aplicar ao cluster. Use o seguinte comando para baixar um exemplo de arquivo.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-full-access-group.yaml
   ```

   Visualizar recursos do Kubernetes em um namespace específico

   O namespace nesse arquivo é `default`, então, se você quiser especificar um namespace diferente, edite o arquivo antes de aplicá-lo ao cluster. Use o comando a seguir para baixar um exemplo de arquivo.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-restricted-access-group.yaml
   ```

2. Edite o arquivo YAML de acesso total ou restrito para substituir as referências de `%IAM_ARN%` pelo nome do recurso da Amazon (ARN) da entidade principal do IAM.
3. Aplique os arquivos YAML de acesso total ou de acesso restrito ao cluster do Kubernetes. Substitua os valores do arquivo YAML por seus próprios valores.

   ```
   kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
   ```

Para ver recursos do Kubernetes no cluster conectado, consulte Visualizar os recursos do Kubernetes. Dados para alguns tipos de recursos na guia Resources (Recursos) não estão disponível para clusters conectados.

Cancelar o registro de um cluster do Kubernetes no console do Amazon EKS

Se você terminou de usar um cluster conectado, pode cancelar seu registro. Depois que o registro for cancelado, o cluster não ficará mais visível no console do Amazon EKS.

É necessário ter estas permissões para chamar a API deregisterCluster:

- `eks:DeregisterCluster`
- `ssm:DeleteActivation`
- `ssm:DeregisterManagedInstance`

Esse processo envolve duas etapas: cancelar o registro do cluster no Amazon EKS e desinstalar o agente do eks-connector do cluster.

Para cancelar o registro do cluster do Kubernetes

AWS CLI

Pré-requisitos

- A AWS CLI deve ser instalada. Para instalar ou atualizar, consulte Instalar a AWS CLI.
- Certifique-se de que a função de agente do Amazon EKS Connector foi criada.

Cancele o registro do cluster conectado.

```
aws eks deregister-cluster \
    --name my-cluster \
    --region region-code
```

AWS Management Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Escolha Clusters.
3. Na página Clusters, selecione o cluster conectado e selecione Deregister (Cancelar registro).
4. Confirme que você deseja cancelar o registro do cluster.

eksctl

Pré-requisitos

- O `eksctl` versão `0.68` ou posterior deve estar instalado. Para obter informações sobre instalação ou upgrade, consulte Conceitos básicos do Amazon EKS: `eksctl`.
- Certifique-se de que a função de agente do Amazon EKS Connector foi criada.

Para cancelar o registro do cluster no **`eksctl`**

- Para a configuração do conector, especifique a função do IAM do agente do Amazon EKS Connector. Para obter mais informações, consulte Funções do IAM necessárias para o Amazon EKS Connector.

```
eksctl deregister cluster --name my-cluster
```

Para limpar os recursos do cluster do Kubernetes

Helm

- Execute o comando a seguir para desinstalar o agente.

```
helm -n eks-connector uninstall eks-connector
```

YAML manifest

1. Exclua o arquivo YAML do Amazon Connector EKS do cluster do Kubernetes.

```
kubectl delete -f eks-connector.yaml
```

2. Se você criou uma `clusterrole` ou `clusterrolebindings` para outras entidades principais do IA visando fornecer acesso ao cluster a mais um usuário do IAM, não se esqueça de excluí-lo do cluster do Kubernetes.

Solução de problemas do Amazon EKS Connector

Este tópico aborda alguns dos erros comuns que você pode encontrar ao usar o Amazon EKS Connector, incluindo instruções sobre como solucioná-los e soluções de contorno.

Solução básica de problemas

Esta seção descreve etapas para diagnosticar o problema se ele não estiver claro.

Verifique o status do Amazon EKS Connector

Verifique o status do Amazon EKS Connector.

```
kubectl get pods -n eks-connector
```

Inspecione logs do Amazon EKS Connector

O Pod do Amazon EKS Connector consiste em três contêineres. Para recuperar logs completos de todos esses contêineres para que possa inspecioná-los, execute os seguintes comandos:

- `connector-init`

```
kubectl logs eks-connector-0 --container connector-init -n eks-connector
kubectl logs eks-connector-1 --container connector-init -n eks-connector
```

- `connector-proxy`

```
kubectl logs eks-connector-0 --container connector-proxy -n eks-connector
kubectl logs eks-connector-1 --container connector-proxy -n eks-connector
```

- `connector-agent`

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
kubectl exec eks-connector-1 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
```

Obtenha o nome efetivo do cluster

Os clusters do Amazon EKS são identificados exclusivamente como `clusterName` em uma única conta da AWS e Região da AWS. Se você tiver vários clusters conectados no Amazon EKS, será possível confirmar em qual cluster do Amazon EKS o cluster atual do Kubernetes está registrado. Para fazê-lo, insira o seguinte para descobrir o `clusterName` do cluster atual.

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
kubectl exec eks-connector-1 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
```

Comandos diversos

Os comandos a seguir são úteis para recuperar informações necessárias para solucionar problemas.

- Use o comando a seguir para coletar imagens usadas pelo Pods no Amazon EKS Connector.

  ```
  kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}"
   | tr -s '[[:space:]]' '\n'
  ```

- Use o comando a seguir para coletar os nomes dos nós nos quais o Amazon EKS Connector está sendo executado.

  ```
  kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" | tr -s
   '[[:space:]]' '\n'
  ```

- Execute o comando a seguir para obter as versões de cliente e servidor do Kubernetes.

  ```
  kubectl version
  ```

- Execute o comando a seguir para obter informações sobre os nós.

  ```
  kubectl get nodes -o wide --show-labels
  ```

Problema do helm: 403 proibido

Se você recebeu o seguinte erro ao executar os comandos de instalação do helm:

```
Error: INSTALLATION FAILED: unexpected status from HEAD request to https://
public.ecr.aws/v2/eks-connector/eks-connector-chart/manifests/0.0.6: 403 Forbidden
```

Você pode executar a seguinte linha para corrigi-lo:

```
docker logout public.ecr.aws
```

Erro do console: o cluster está paralisado no estado Pending (Pendente)

Se o cluster ficar paralisado no estado `Pending` no console do Amazon EKS depois que você o registrou, isso pode indicar que o Amazon EKS Connector ainda não conectou o cluster à AWS com êxito. Em um cluster registrado, o estado `Pending` significa que a conexão não foi estabelecida com êxito. Para solucionar esse problema, certifique-se de que aplicou o manifesto ao cluster do Kubernetes de destino. Se você o aplicou ao cluster, mas o cluster permanece no estado `Pending`, pode ser que o statefulset `eks-connector` não esteja íntegro. Para solucionar esse problema, consulte Os Pods do Amazon EKS Connector estão em um loop de pane neste tópico.

Erro do console: **`User “system:serviceaccount:eks-connector:eks-connector” can't impersonate resource “users” in API group “”`** no escopo do cluster

O Amazon EKS Connector usa a representação do usuário do Kubernetes para atuar em nome das entidades principais do IAM no AWS Management Console. Toda entidade principal que acessa a API do Kubernetes na conta de serviço `eks-connector` da AWS precisa receber uma permissão para representar o usuário do Kubernetescorrespondente usando um ARN do IAM como o nome de usuário do Kubernetes. Nos exemplos a seguir, o ARN do IAM é mapeado para um usuário do Kubernetes.

- O usuário do IAM *`john`*, da conta da AWS *`111122223333`* é mapeado para um usuário do Kubernetes. As melhores práticas do IAM recomendam que você conceda permissões para perfis e não para usuários.

  ```
  arn:aws:iam::111122223333:user/john
  ```

- O perfil do IAM de *`admin`* da conta da AWS *`111122223333`* é mapeado para um usuário do Kubernetes:

```
arn:aws:iam::111122223333:role/admin
```

O resultado é um perfil do IAM, em vez do ARN da sessão do AWS STS.

Para obter instruções sobre como configurar o `ClusterRole` e o `ClusterRoleBinding` para conceder à conta de serviço do `eks-connector` o privilégio de representar o usuário mapeado, consulte Conceder acesso para visualizar recursos de clusters do Kubernetes em um console do Amazon EKS. Substitua `%IAM_ARN%` no modelo pelo ARN do IAM do AWS Management Console da entidade principal do IAM.

Erro do console: **`[...] is forbidden: User [...] cannot list resource "[...] in API group"`** no escopo do cluster

Considere o seguinte problema. O Amazon EKS Connector representou com êxito a entidade principal do IAM no AWS Management Console da solicitação no cluster do Kubernetes de destino. Porém, a entidade principal representada não tem as permissões RBAC para as operações de API do Kubernetes.

Para resolver esse problema, existem dois métodos de conceder permissões a usuários adicionais. Se você instalou anteriormente o eks-connector por meio do chart do Helm, poderá conceder acesso aos usuários facilmente executando o comando a seguir. Substitua o `userARN1` e o `userARN2` por uma lista dos ARNs dos perfis do IAM para conceder acesso para visualização dos recursos do Kubernetes:

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
    --reuse-values \
    --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

Ou, como administrador do cluster, conceda aos usuários individuais do Kubernetes o nível apropriado de privilégios de RBAC. Para ter mais informações e exemplos, consulte Conceder acesso para visualizar recursos de clusters do Kubernetes em um console do Amazon EKS.

Erro do console: O Amazon EKS não consegue se comunicar com o Kubernetes servidor de API do cluster. O cluster deve estar em um estado ATIVO para estabelecer uma conexão bem-sucedida. Tente novamente em alguns minutos.

Se o serviço Amazon EKS não conseguir se comunicar com o Amazon EKS Connector no cluster de destino, pode ser por um dos seguintes motivos:

- O Amazon EKS Connector no cluster de destino não está íntegro.
- Baixa conectividade ou conexão interrompida entre o cluster de destino e a Região da AWS.

Para resolver esse problema, verifique os logs do Amazon EKS Connector. Se você não encontrar nenhum erro para o Amazon EKS Connector, tente fazer a conexão novamente após alguns minutos. Se você enfrentar regularmente alta latência ou conectividade intermitente para o cluster de destino, considere registrá-lo novamente em uma Região da AWS mais perto de você.

Os Pods do Amazon EKS Connector estão em um loop de pane

Muitos motivos podem fazer um Pod do Amazon EKS Connector entrar no status `CrashLoopBackOff`. É provável que esse problema envolva o contêiner `connector-init`. Verifique o status do Pod do Amazon EKS Connector.

```
kubectl get pods -n eks-connector
```

Veja um exemplo de saída abaixo.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:CrashLoopBackOff   1          7s
```

Se a saída for semelhante à saída anterior, consulte Inspecione logs do Amazon EKS Connector para solucionar o problema.

`Failed to initiate eks-connector: InvalidActivation`

Quando você inicia o Amazon EKS Connector pela primeira vez, ele registra um `activationId` e um `activationCode` na Amazon Web Services. O registro pode falhar, o que pode fazer com que o contêiner `connector-init` falhe com um erro similar ao erro a seguir.

```
F1116 20:30:47.261469       1 init.go:43] failed to initiate eks-connector:
 InvalidActivation:
```

Para solucionar esse problema, considere as seguintes causas e correções recomendadas:

- O registro pode ter falhado porque o `activationId` e o `activationCode` não estavam no arquivo de manifesto. Se for esse o caso, verifique se os valores corretos retornados na operação da API `RegisterCluster` estavam corretos e se o `activationCode` está no arquivo de manifesto. O `activationCode` é adicionado aos segredos do Kubernetes, portanto, deve ser codificado em `base64`. Para obter mais informações, consulte Etapa 1: Registrar o cluster.
- O registro pode ter falhado porque sua ativação expirou. Isso ocorre porque, por motivos de segurança, você deve ativar o Amazon EKS Connector em até três dias após registrar o cluster. Para resolver esse problema, certifique-se de que o manifesto do Amazon EKS Connector seja aplicado ao cluster do Kubernetes de destino antes da data e hora de expiração. Para confirmar a data de expiração da ativação, chame a operação da API `DescribeCluster`.

  ```
  aws eks describe-cluster --name my-cluster
  ```

  No exemplo de resposta a seguir, a data e a hora de expiração estão registradas como `2021-11-12T22:28:51.101000-08:00`.

  ```
  {
      "cluster": {
          "name": "my-cluster",
          "arn": "arn:aws:eks:region:111122223333:cluster/my-cluster",
          "createdAt": "2021-11-09T22:28:51.449000-08:00",
          "status": "FAILED",
          "tags": {
          },
          "connectorConfig": {
              "activationId": "00000000-0000-0000-0000-000000000000",
              "activationExpiry": "2021-11-12T22:28:51.101000-08:00",
              "provider": "OTHER",
              "roleArn": "arn:aws:iam::111122223333:role/my-connector-role"
          }
      }
  }
  ```

Se a `activationExpiry` já tiver passado, cancele o registro do cluster e registre-o novamente. Isso gerará uma nova ativação.

O nó do cluster está sem conectividade de saída

Para funcionar corretamente, o Amazon EKS Connector requer conectividade de saída para vários endpoints da AWS. Não é possível conectar um cluster privado sem conectividade de saída a uma Região da AWS de destino. Para solucionar esse problema, você deve adicionar a conectividade de saída necessária. Para obter mais informações sobre requisitos de conector, consulte Considerações sobre o Amazon EKS Connector.

O Amazon EKS Connector Pods está no estado **`ImagePullBackOff`**

Se você executar o comando `get pods` e os Pods estiverem no estado `ImagePullBackOff`, não poderão funcionar corretamente. Se os Pods do Amazon EKS Connector estiverem no estado `ImagePullBackOff`, não poderão funcionar corretamente. Verifique o status dos Pods do Amazon EKS Connector.

```
kubectl get pods -n eks-connector
```

Veja um exemplo de saída abaixo.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:ImagePullBackOff   0          4s
```

O arquivo de manifesto padrão do Amazon EKS Connector referencia imagens da Amazon ECR Public Gallery. É possível que o cluster do Kubernetes de destino não possa extrair imagens da Amazon ECR Public Gallery. Solucione o problema de extração de imagens da Amazon ECR Public Gallery ou considere espelhar as imagens no registro do contêiner privado de sua escolha.

Perguntas frequentes sobre o AWS Connector

P: Como funciona a tecnologia subjacente por trás do Amazon EKS Connector?

R: O Amazon EKS Connector é baseado no agente do AWS Systems Manager (Systems Manager). O Amazon EKS Connector é executado como um `StatefulSet` no cluster do Kubernetes. Ele estabelece uma conexão e configura o proxy da comunicação entre o servidor da API do cluster e a

Amazon Web Services. Ele faz isso para exibir dados de cluster no console do Amazon EKS até que você desconecte o cluster da AWS. O agente do Systems Manager é um projeto de código aberto. Para obter mais informações sobre o projeto, consulte a GitHubpágina do projeto no GitHub.

P: Tenho um cluster on-premises do Kubernetes que quero conectar. Preciso abrir portas de firewall para conectá-lo?

R: Não, não é necessário abrir nenhuma porta de firewall. O cluster do Kubernetes exige apenas conexão de saída com a Regiões da AWS. Os serviços da AWS nunca acessam recursos na rede on-premises. O Amazon EKS Connector é executado no seu cluster e inicia a conexão com a AWS. Quando o registro do cluster é concluído, a AWS só emite comandos para o Amazon EKS Connector depois que você inicia uma ação no console do Amazon EKS que exija informações do servidor de API do Kubernetes no cluster.

P: Quais dados são enviados do meu cluster para a AWS pelo Amazon EKS Connector?

R: O Amazon EKS Connector envia informações técnicas necessárias para o registro do seu cluster na AWS. Ele também envia metadados de clusters e workloads para os recursos do console do Amazon EKS que os clientes solicitam. O Amazon EKS Connector só coletará ou enviará esses dados se você iniciar uma ação no console ou na API do Amazon EKS que exija que os dados sejam enviados para a AWS. Exceto pelo número da versão do Kubernetes, a AWS não armazena dados por padrão. Ele só armazenará os dados se você autorizar.

P: Posso conectar um cluster fora de uma Região da AWS?

R: Sim, você pode conectar um cluster de qualquer local ao Amazon EKS. Além disso, o serviço Amazon EKS pode estar localizado em qualquer Região da AWS comercial pública da AWS. Isso funciona com uma conexão de rede válida do seu cluster para a Região da AWS de destino. Recomendamos que você escolha uma Região da AWS que seja mais próxima da localização do cluster para otimizar a performance da interface do usuário. Por exemplo, se você tiver um cluster em execução em Tóquio, conecte o seu cluster à Região da AWS em Tóquio (isto é, a Região da AWS `ap-northeast-1`) para obter baixa latência. Você pode conectar um cluster de qualquer local ao Amazon EKS em qualquer uma das Regiões da AWS comerciais públicas, exceto as Regiões da AWS da China ou GovCloud.

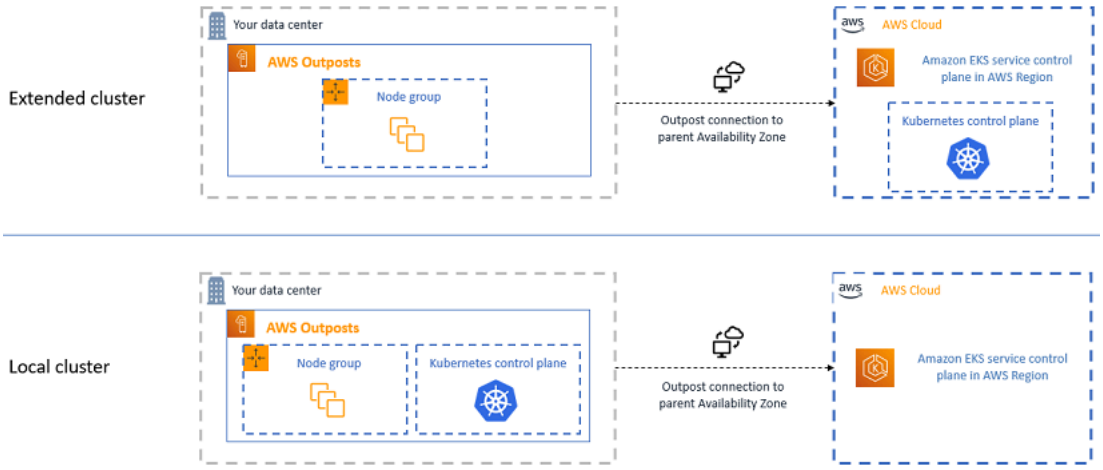
Implantar o Amazon EKS on-premises com o AWS Outposts

É possível usar o Amazon EKS para executar aplicações do Kubernetes on-premises no AWS Outposts. É possível implantar o Amazon EKS no Outposts das seguintes maneiras:

- Clusters estendidos: executar o ambiente de gerenciamento do Kubernetes em uma Região da AWS e em nós no Outpost.
- Clusters locais: executar o ambiente de gerenciamento do Kubernetes no Outpost.

Para ambas as opções de implantação, o ambiente de gerenciamento do Kubernetes é totalmente gerenciado pela AWS. É possível utilizar as mesmas APIs, ferramentas e console do Amazon EKS utilizados na nuvem para criar e executar o Amazon EKS no Outposts.

O diagrama a seguir mostra essas opções de implantação.



Quando usar cada opção de implantação

Os clusters locais e estendidos são opções de implantação de uso geral e podem ser usados para uma variedade de aplicações.

Os clusters locais permitem que você execute todo o cluster do Amazon EKS localmente no Outposts. Isso ajuda a reduzir o risco de tempo de inatividade das aplicações que pode resultar de perdas temporárias de conexão da rede com a nuvem. Essas desconexões da rede podem ser causadas por cortes de fibra ou eventos climáticos. Como todo o cluster do Amazon EKS é executado localmente no Outposts, as aplicações permanecem disponíveis. Você pode executar operações de cluster durante as desconexões da rede com a nuvem. Para obter mais informações,

consulte Preparar clusters locais do Amazon EKS no AWS Outposts para desconexões de rede. Se você tiver preocupações com a qualidade da conexão de rede do Outposts com a Região da AWS superior e precisar de alta disponibilidade durante desconexões da rede, use a opção de implantação de cluster local.

Os clusters estendidos permitem que você conserve a capacidade no Oupost porque o ambiente de gerenciamento do Kubernetes é executado na Região da AWS superior. Essa opção pode ser mais adequada se você puder investir em conectividade de rede confiável e redundante do Outpost com a Região da AWS. A qualidade da conexão de rede é fundamental para essa opção. A forma como o Kubernetes lida com as desconexões da rede entre o ambiente de gerenciamento do Kubernetes e os nós pode causar tempo de inatividade das aplicações. Para obter mais informações sobre o comportamento do Kubernetes, consulte Programação, preempção e remoção na documentação do Kubernetes.

Comparar as opções de implantação

A tabela a seguir compara as diferenças entre as duas opções.

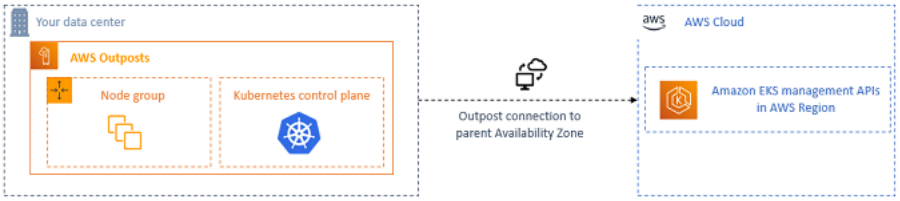
Atributo	Cluster estendido	Cluster local
Local do ambiente de gerenciamento do Kubernetes	Região da AWS	Outpost
Conta do ambiente de gerenciamento do Kubernetes	Conta da AWS	Sua conta
Disponibilidade regional	Consulte Endpoints de serviço	Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia) , Oeste dos EUA (N. da Califórnia), Oeste dos EUA (Oregon), Ásia-Pacífico (Seul), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio) Canadá (Central), Europa (Frankfur t), Europa (Irlanda), Europa (Londres), Oriente Médio

Atributo	Cluster estendido	Cluster local
		(Bahrein) e América do Sul (São Paulo)
Versões secundárias do Kubernetes	Versões do Amazon EKS com suporte.	Versões do Amazon EKS com suporte.
Versões da plataforma	Consulte Veja as versões da plataforma do Amazon EKS para cada versão do Kubernetes	Consulte Conheça as versões de plataforma do Kubernetes e do Amazon EKS para AWS Outposts
Fatores de formato do Outpost	Racks do Outpost	Racks do Outpost
Interfaces do usuário	AWS Management Console, AWS CLI, API do Amazon EKS, `eksctl`, AWS CloudFormation e Terraform	AWS Management Console, AWS CLI, API do Amazon EKS, `eksctl`, AWS CloudFormation e Terraform
Políticas gerenciadas	AmazonEKSClusterPolicy e AmazonEKSServiceRolePolicy	AmazonEKSLocalOutpostClusterPolicy e AmazonEKSLocalOutpostServiceRolePolicy
VPC e sub-redes de clusters	Consulte Requisitos e considerações sobre a VPC e a sub-rede do Amazon EKS	Consulte Criar uma VPC e sub-redes para clusters do Amazon EKS no AWS Outposts
Acesso ao endpoint do cluster	Público ou privado ou ambos	Privado apenas
Autenticação do servidor de API do Kubernetes	AWS Identity and Access Management (IAM) e OIDC	IAM e certificados `x.509`
Tipos de nó	Somente autogerenciado	Somente autogerenciado
Tipos de computação de nós	Amazon EC2 sob demanda	Amazon EC2 sob demanda

Atributo	Cluster estendido	Cluster local
Tipos de armazenamento de nós	Amazon EBS gp2 e SSD de NVMe local	Amazon EBS gp2 e SSD de NVMe local
AMIs otimizadas para o Amazon EKS	Amazon Linux, Windows e Bottlerocket	Somente para Amazon Linux
Versões de IP	Somente `IPv4`	Somente `IPv4`
Complementos	Complementos do Amazon EKS ou complementos autogerenciados	Somente complementos autogerenciados
Container Network Interface padrão	Amazon VPC CNI plugin for Kubernetes	Amazon VPC CNI plugin for Kubernetes
Logs do ambiente de gerenciamento do Kubernetes	Amazon CloudWatch Logs	Amazon CloudWatch Logs
Balanceamento de carga	Use o AWS Load Balancer Controller para provisionar somente Application Load Balancers (sem Network Load Balancers)	Use o AWS Load Balancer Controller para provisionar somente Application Load Balancers (sem Network Load Balancers)
Criptografia envelopada de segredos	Consulte Criptografar segredos do Kubernetes com o AWS KMS em clusters existentes	Sem compatibilidade
Perfis do IAM para contas de serviço	Consulte Perfis do IAM para contas de serviço	Sem compatibilidade
Solução de problemas	Consulte Solucionar problemas com clusters e nós do Amazon EKS	Consulte Solucionar problemas de clusters locais do Amazon EKS no AWS Outposts

Criar clusters locais do Amazon EKS no AWS Outposts para garantir alta disponibilidade

Você pode usar clusters locais para executar todo o cluster do Amazon EKS localmente no AWS Outposts. Isso ajuda a reduzir o risco de tempo de inatividade da aplicação, que pode resultar de desconexões temporárias da rede com a nuvem. Essas desconexões podem ser causadas por cortes de fibra ou eventos climáticos. Como o cluster inteiro do Kubernetes é executado localmente no Outposts, as aplicações permanecem disponíveis. Você pode executar operações de cluster durante as desconexões da rede com a nuvem. Para obter mais informações, consulte Preparar clusters locais do Amazon EKS no AWS Outposts para desconexões de rede. O diagrama a seguir mostra a implantação de um cluster local.



Clusters locais geralmente estão disponíveis para uso com racks do Outposts.

Com suporte Regiões da AWS

Você pode criar clusters locais nas seguintes Regiões da AWS: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (N. da Califórnia), Oeste dos EUA (Oregon), Ásia-Pacífico (Seul), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central), Europa (Frankfurt), Europa (Irlanda), Europa (Londres), Oriente Médio (Bahrein) e América do Sul (São Paulo). Para obter informações sobre os recursos compatíveis, consulte Comparar as opções de implantação.

Tópicos

- Implantar um cluster do Amazon EKS no AWS Outposts
- Conheça as versões de plataforma do Kubernetes e do Amazon EKS para AWS Outposts
- Criar uma VPC e sub-redes para clusters do Amazon EKS no AWS Outposts
- Preparar clusters locais do Amazon EKS no AWS Outposts para desconexões de rede
- Selecionar tipos de instância e grupos de posicionamento para clusters do Amazon EKS no AWS Outposts com base em considerações de capacidade
- Solucionar problemas de clusters locais do Amazon EKS no AWS Outposts

Implantar um cluster do Amazon EKS no AWS Outposts

Este tópico fornece uma visão geral do que deve ser considerado ao executar um cluster local em um Outpost. O tópico também fornece instruções sobre como implantar um cluster local em um Outpost.

Considerações

> **Important**
>
> - Essas considerações não são replicadas na documentação relacionada do Amazon EKS. Se outros tópicos da documentação do Amazon EKS entrarem em conflito com as considerações aqui apresentadas, siga estas considerações.
> - Essas considerações estão sujeitas a alterações, que podem ocorrer com frequência. Portanto, recomendamos que você revise este tópico periodicamente.
> - Muitas das considerações são diferentes das considerações para a criação de um cluster na Nuvem AWS.

- Os clusters locais somente são compatíveis com racks do Outpost. Um único cluster local pode ser executado em vários racks físicos do Outpost que incluem um único Outpost lógico. Um único cluster local não pode ser executado em vários Outposts lógicos. Cada Outpost lógico tem um único ARN de Outpost.
- Clusters locais executam e gerenciam o ambiente de gerenciamento do Kubernetes na sua conta no Outpost. Você não pode executar workloads nas instâncias do ambiente de gerenciamento do Kubernetes nem modificar os componentes do ambiente de gerenciamento do Kubernetes. Esses nós são gerenciados pelo serviço do Amazon EKS. Alterações no ambiente de gerenciamento do Kubernetes não persistem por meio de ações automáticas de gerenciamento do Amazon EKS, como aplicação de patches.
- Os clusters locais são compatíveis com complementos autogerenciados e grupos de nós autogerenciados do Amazon Linux. Os complementos Amazon VPC CNI plugin for Kubernetes, `kube-proxy` e CoreDNS são instalados automaticamente nos clusters locais.
- Os clusters locais exigem o uso do Amazon EBS no Outposts. Seu Outpost deve ter o Amazon EBS disponível para o armazenamento do ambiente de gerenciamento do Kubernetes.

- Os clusters usam o Amazon EBS no Outposts. Seu Outpost deve ter o Amazon EBS disponível para o armazenamento do ambiente de gerenciamento do Kubernetes. O Outposts somente é compatível com volumes gp2 do Amazon EBS.
- Kubernetes `PersistentVolumes` baseados no Amazon EBS são compatíveis com o uso do driver da CSI do Amazon EBS.

Pré-requisitos

- Familiaridade com as Opções de implantação do Outposts, Selecionar tipos de instância e grupos de posicionamento para clusters do Amazon EKS no AWS Outposts com base em considerações de capacidade e Criar uma VPC e sub-redes para clusters do Amazon EKS no AWS Outposts.
- Um Outpost existente. Para obter mais informações, consulte O que é o AWS Outposts.
- A ferramenta da linha de comando `kubectl` instalada no seu computador ou no AWS CloudShell. A versão pode ser idêntica ou até uma versão secundária anterior ou posterior à versão Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte Configurar o `kubectl` e o `eksctl`.
- A versão `2.12.3` ou superior ou a versão `1.27.160` ou superior da AWS Command Line Interface (AWS CLI) instalada e configurada em seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**. Gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI e Configuração rápida com o aws configure no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-la, consulte Instalar a AWS CLI no diretório inicial no Guia do usuário do AWS CloudShell.
- Uma entidade principal do IAM (usuário ou perfil) com permissões para `create` e `describe` um cluster do Amazon EKS. Para ter mais informações, consulte Criar um cluster local do Kubernetes em um Outpost e Listar ou descrever todos os clusters.

Quando um cluster local do Amazon EKS é criado, a entidade principal do IAM que cria o cluster é adicionada permanentemente. A entidade principal é adicionada especificamente à tabela de autorização RBAC do Kubernetes como administradora. Esta entidade tem permissões `system:masters`. A identidade desta entidade não fica visível na configuração do seu cluster. Por isso, é importante observar a entidade que criou o cluster e garantir que ela nunca tenha sido

excluída. Inicialmente, somente a entidade principal que criou o servidor poderá fazer chamadas para o servidor de API do Kubernetes usando o `kubectl`. Se usar o console para criar o cluster, você deverá se certificar de que as mesmas credenciais do IAM estejam na cadeia de credenciais do AWS SDK quando executar os comandos `kubectl` no cluster. Depois de criar o cluster, você poderá conceder acesso a ele para outras entidades do IAM.

Para criar um cluster local do Amazon EKS

É possível criar um cluster local com `eksctl`, com o AWS Management Console, com a AWS CLI, com a API do Amazon EKS, com SDKs da AWS, com o AWS CloudFormation ou com o Terraform.

1. Crie um cluster local.

   eksctl

   Pré-requisito

   Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

   Como criar o cluster com o **`eksctl`**

   1. Copie o conteúdo a seguir no seu dispositivo. Substitua os valores a seguir e execute o comando modificado para criar o arquivo `outpost-control-plane.yaml`:
      - Substitua *`region-code`* pela Região da AWS compatível na qual você deseja criar o cluster.
      - Substitua *`my-cluster`* por um nome de cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
      - Substitua *`vpc-ExampleID1`* e *`subnet-ExampleID1`* pelos IDs da sua VPC e da sub-rede existentes. A VPC e a sub-rede devem atender aos requisitos em Criar uma VPC e sub-redes para clusters do Amazon EKS no AWS Outposts.
      - Substitua *`uniqueid`* pelo ID do seu Outpost.

- Substitua *m5.large* por um tipo de instância disponível em seu Outpost. Antes de escolher um tipo de instância, consulte Selecionar tipos de instância e grupos de posicionamento para clusters do Amazon EKS no AWS Outposts com base em considerações de capacidade. Três instâncias do ambiente de gerenciamento são implantadas. Não é possível alterar esse número.

```
cat >outpost-control-plane.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "1.24"

vpc:
  clusterEndpoints:
    privateAccess: true
  id: "vpc-vpc-ExampleID1"
  subnets:
    private:
      outpost-subnet-1:
        id: "subnet-subnet-ExampleID1"

outpost:
  controlPlaneOutpostARN: arn:aws:outposts:region-code:111122223333:outpost/
op-uniqueid
  controlPlaneInstanceType: m5.large
EOF
```

Para obter uma lista completa de todas as opções e padrões disponíveis, consulte Suporte do AWS Outposts e Esquema do arquivo Config na documentação do `eksctl`.

2. Crie o cluster usando o arquivo de configuração que você criou na etapa anterior. O `eksctl` cria uma VPC e uma sub-rede no seu Outpost para implantar o cluster.

```
eksctl create cluster -f outpost-control-plane.yaml
```

O provisionamento de cluster leva alguns minutos. Durante a criação do cluster, várias linhas de saída são exibidas. A última linha do resultado é semelhante ao exemplo de linha a seguir.

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> Para ver a maioria das opções que podem ser especificadas ao criar um cluster com `eksctl`, use o comando **`eksctl create cluster --help`**. Para ver todas as opções disponíveis, é possível utilizar um arquivo `config`. Para obter mais informações, consulte Uso dos arquivos de configuração e o esquema de arquivo de configuração na documentação do `eksctl`. Você pode encontrar exemplos de arquivos de configuração no GitHub.

`Eksctl` criou automaticamente uma entrada de acesso para a entidade principal do IAM (usuário ou perfil) que criou o cluster e concedeu ao administrador principal do IAM permissões para objetos do Kubernetes no cluster. Se não quiser que o criador do cluster tenha acesso de administrador aos objetos do Kubernetes no cluster, adicione o seguinte texto ao arquivo de configuração anterior: **`bootstrapClusterCreatorAdminPermissions: false`** (no mesmo nível de `metadata`, `vpc` e `outpost`). Se você adicionou a opção, então, depois da criação do cluster, precisará criar uma entrada de acesso para pelo menos uma entidade principal do IAM, ou nenhuma entidade principal do IAM terá acesso aos objetos do Kubernetes no cluster.

AWS Management Console

Pré-requisito

Uma VPC e sub-redes existentes que atendam a requisitos do Amazon EKS. Para obter mais informações, consulte Criar uma VPC e sub-redes para clusters do Amazon EKS no AWS Outposts.

Para criar o cluster com a AWS Management Console

1. Se você já tiver um perfil do IAM do cluster local ou se for criar seu cluster com `eksctl`, poderá ignorar essa etapa. Por padrão, `eksctl` cria um perfil para você.
   a. Execute o seguinte comando para criar um arquivo JSON de política de confiança do IAM:

```
cat >eks-local-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. Crie o perfil do IAM do cluster do Amazon EKS. Para criar um perfil do IAM, a entidade principal do IAM que estiver criando o perfil deverá ser atribuída à seguinte ação `iam:CreateRole` (permissão):

```
aws iam create-role --role-name myAmazonEKSLocalClusterRole --assume-role-
policy-document file://"eks-local-cluster-role-trust-policy.json"
```

c. Anexe a política gerenciada do Amazon EKS denominada `AmazonEKSLocalOutpostClusterPolicy` à função. Para anexar uma política do IAM a uma entidade principal do IAM a entidade principal do IAM que está anexando a política deve receber uma das seguintes ações do IAM (permissões): `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

2. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
3. Na parte superior da tela do console, certifique-se de ter selecionado uma Região da AWS compatível.
4. Escolha Add cluster (Adicionar cluster) e, em seguida, Create (Criar).
5. Na página Configure cluster (Configurar cluster), insira ou selecione valores para os seguintes campos:
   - Localização do ambiente de gerenciamento do Kubernetes – Escolha AWS Outposts.

- Outpost ID (ID do Outpost): escolha o ID do Outpost no qual você deseja criar seu ambiente de gerenciamento.
- Em Instance type (Tipo de instância): selecione um tipo de instância. Somente os tipos de instância disponíveis no seu Outpost são exibidos. Na lista suspensa, cada tipo de instância descreve para quantos nós o tipo de instância é recomendado. Antes de escolher um tipo de instância, consulte Selecionar tipos de instância e grupos de posicionamento para clusters do Amazon EKS no AWS Outposts com base em considerações de capacidade. Todas as réplicas são implantadas com o uso do mesmo tipo de instância. Você não poderá alterar o tipo de instância depois que seu cluster for criado. Três instâncias do ambiente de gerenciamento são implantadas. Não é possível alterar esse número.
- Name (Nome): um nome exclusivo para o cluster. Ele deve ser exclusivo na sua Conta da AWS. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
- Versão do Kubernetes: escolha a versão do Kubernetes que deseja usar para o cluster. A menos que precise usar uma versão anterior, recomendamos que você selecione a versão mais recente.
- Perfil de serviço do cluster: escolha o perfil do IAM do cluster do Amazon EKS criado em uma etapa anterior para permitir que o ambiente de gerenciamento do Kubernetes gerencie os recursos da AWS.
- Acesso de administrador do cluster do Kubernetes: se você quiser que a entidade principal do IAM (perfil ou usuário) que está criando o cluster tenha acesso de administrador aos objetos do Kubernetes no cluster, aceite o padrão (permitir). O Amazon EKS cria uma entrada de acesso para a entidade principal do IAM e concede permissões de administrador do cluster à entrada de acesso. Para obter mais informações sobre as entradas de acesso, consulte Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS.

  Se você quiser que uma entidade principal do IAM diferente daquela que está criando o cluster tenha acesso de administrador aos objetos do Kubernetes do cluster, escolha a opção de não permitir. Após a criação do cluster, qualquer entidade principal do IAM que tenha permissões do IAM para criar entradas de acesso poderá adicionar uma entrada de acesso para qualquer entidade principal do IAM que precise acessar os

objetos do Kubernetes do cluster. Para obter mais informações sobre as permissões necessárias do IAM, consulte Ações definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço. Se você escolher a opção de não permitir e não criar nenhuma entrada de acesso, nenhuma entidade principal do IAM terá acesso aos objetos do Kubernetes no cluster.

- Tags (Etiquetas) – (opcional) adicione etiquetas ao cluster. Para obter mais informações, consulte Organizar recursos do Amazon EKS com tags.

Quando terminar com essa página, escolha Próximo.

6. Na página Specify networking (Especificar redes), selecione valores para os seguintes campos:
   - VPC: escolha uma VPC existente. A VPC deverá ter uma quantidade suficiente de endereços IP disponíveis para o cluster, todos os nós e outros recursos do Kubernetes que você desejar criar. A VPC deve atender aos requisitos em Requisitos e considerações para VPCs.
   - Subnets (Sub-redes): por padrão, as sub-redes disponíveis na VPC especificada no campo anterior são pré-selecionadas. As sub-redes escolhidas devem atender aos requisitos de Requisitos e considerações para sub-redes.

     Security groups (Grupos de segurança) (Opcional): especifique um ou mais grupos de segurança que você deseja que o Amazon EKS associe às interfaces de rede que ele cria. O Amazon EKS cria automaticamente um grupo de segurança que permite a comunicação entre o cluster e a VPC. O Amazon EKS associa esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte Considerações e requisitos sobre grupos de segurança do Amazon EKS. É possível modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS. Se optar por adicionar seus próprios grupos de segurança, você não poderá alterar os escolhidos após a criação desse cluster. Para que os hosts on-premises se comuniquem com o endpoint do cluster, você deve permitir tráfego de entrada do grupo de segurança do cluster. Para clusters que não têm uma conexão de internet de entrada e saída (também conhecidos como clusters privados), você deve fazer o seguinte:
     - Adicione o grupo de segurança associado aos endpoints da VPC necessários. Para obter mais informações sobre os endpoints necessários, consulte Endpoints da VCP de interface em Acesso à sub-rede para Serviços da AWS.

- Modifique o grupo de segurança que o Amazon EKS criou para permitir tráfego do grupo de segurança associado aos endpoints da VPC.

Quando terminar com essa página, escolha Próximo.

7. Na página Configurar observabilidade, é possível escolher opcionalmente quais opções de Métricas e Log do ambiente de gerenciamento você deseja ativar. Por padrão, o cada tipo de log está desativado.
   - Para obter mais informações sobre as opções de métricas do Prometheus, consulte Etapa 1: ativar as métricas do Prometheus ao criar um cluster.
   - Para obter mais informações sobre as opções do Log do ambiente de gerenciamento, consulte Enviar logs do ambiente de gerenciamento para o CloudWatch Logs.

   Quando terminar com essa página, escolha Próximo.
8. Na página Review and create (Revisar e criar), revise as informações que você inseriu ou selecionou nas páginas anteriores. Se precisar fazer alterações, escolha Edit (Editar). Quando estiver satisfeito, escolha Create (Criar). O campo Status mostra o campo CREATING (Criando) enquanto o cluster é provisionado.

   O provisionamento de cluster leva alguns minutos.

2. Depois que o cluster for criado, você poderá visualizar as instâncias do ambiente de gerenciamento do Amazon EC2 que foram criadas.

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?
Key==`Name`]|[0].Value}' | grep my-cluster-control-plane
```

Veja um exemplo de saída abaixo.

```
"Name": "my-cluster-control-plane-id1"
"Name": "my-cluster-control-plane-id2"
"Name": "my-cluster-control-plane-id3"
```

Cada instância recebeu taints com o `node-role.eks-local.amazonaws.com/control-plane` para que nenhuma workload seja programada nas instâncias do ambiente de gerenciamento. Para obter mais informações sobre taints, consulte Taints e tolerâncias na documentação do Kubernetes. O Amazon EKS monitora continuamente o estado dos clusters locais. Executamos ações automáticas de gerenciamento, como patches de segurança e reparo de instâncias não íntegras. Quando os clusters locais são desconectados da nuvem,

executamos ações para garantir que o cluster seja reparado para um estado íntegro após a reconexão.

3. Se você criou seu cluster usando `eksctl`, pode ignorar esta etapa. O `eksctl` completa esta etapa para você. Habilite o `kubectl` para se comunicar com o cluster adicionando um novo contexto ao arquivo `kubectl config`. Para obter instruções sobre como criar e atualizar o arquivo, consulte Conectar o kubectl a um cluster do EKS criando um arquivo kubeconfig.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Veja um exemplo de saída abaixo.

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

4. Para se conectar ao servidor de API do Kubernetes do cluster local, você deve ter acesso ao gateway local da sub-rede ou conectar-se de dentro da VPC. Para obter mais informações sobre a conexão de um rack do Outpost à sua rede on-premises, consulte How local gateways for racks work (Como funcionam os gateways locais para racks) no Guia do usuário do AWS Outposts. Se você usar roteamento direto de VPC e a sub-rede do Outpost tiver uma rota para o gateway local, os endereços IP privados das instâncias do ambiente de gerenciamento do Kubernetes serão transmitidos automaticamente pela rede local. O endpoint do servidor de API do Kubernetes do cluster local é hospedado no Amazon Route 53 (Route 53). O endpoint do serviço de API pode ser resolvido por servidores DNS públicos para os endereços IP privados dos servidores de API do Kubernetes.

   As instâncias do ambiente de gerenciamento do Kubernetes dos clusters locais são configuradas com interfaces de rede elásticas estáticas com endereços IP privados fixos que não mudam durante o ciclo de vida do cluster. As máquinas que interagem com o servidor de API do Kubernetes podem não ter conectividade com o Route 53 durante as desconexões da rede. Se for esse o caso, recomendamos configurar `/etc/hosts` com os endereços IP privados estáticos para operações contínuas. Também recomendamos configurar servidores DNS locais e conectá-los ao Outpost. Para obter mais informações, consulte a documentação do AWS Outposts. Execute o comando a seguir para confirmar que a comunicação foi estabelecida com o cluster.

```
kubectl get svc
```

Veja um exemplo de saída abaixo.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

5. (Opcional) Teste a autenticação no cluster local quando ele estiver em um estado desconectado da Nuvem AWS. Para obter instruções, consulte Preparar clusters locais do Amazon EKS no AWS Outposts para desconexões de rede.

Recursos internos

O Amazon EKS cria os seguintes recursos no seu cluster. Os recursos são para uso interno do Amazon EKS. Para o funcionamento adequado do cluster, não edite nem modifique esses recursos.

- Os seguintes Pods de espelho:
  - `aws-iam-authenticator-`*`node-hostname`*
  - `eks-certificates-controller-`*`node-hostname`*
  - `etcd-`*`node-hostname`*
  - `kube-apiserver-`*`node-hostname`*
  - `kube-controller-manager-`*`node-hostname`*
  - `kube-scheduler-`*`node-hostname`*
- Os seguintes complementos autogerenciados:
  - `kube-system/coredns`
  - `kube-system/kube-proxy` (só será criado quando você adicionar seu primeiro nó)
  - `kube-system/aws-node` (só será criado quando você adicionar seu primeiro nó). Clusters locais usam o plugin Amazon VPC CNI plugin for Kubernetes para redes de cluster. Não altere a configuração das instâncias do ambiente de gerenciamento (pods denominados `aws-node-controlplane-*`). Há variáveis de configuração que você pode usar para alterar o valor padrão para quando o plug-in criar novas interfaces de rede. Para obter mais informações, consulte a documentação sobre o GitHub.
- Os seguintes serviços:
  - `default/kubernetes`
  - `kube-system/kube-dns`
- Um `PodSecurityPolicy` denominado `eks.system`

- Um `ClusterRole` denominado `eks:system:podsecuritypolicy`
- Um `ClusterRoleBinding` denominado `eks:system`
- Um PodSecurityPolicy padrão
- Além do grupo de segurança do cluster, o Amazon EKS cria um grupo de segurança na Conta da AWS denominado `eks-local-internal-do-not-use-or-edit-`*`cluster-name`*`-`*`uniqueid`*. Esse grupo de segurança permite que o tráfego flua livremente entre componentes do Kubernetes em execução nas instâncias do ambiente de gerenciamento.

Próximas etapas recomendadas:

- Conceda à entidade principal do IAM que criou o cluster as permissões necessárias para visualizar os recursos do Kubernetes no AWS Management Console
- Conceda acesso para entidades do IAM ao cluster. Se quiser que as entidades visualizem os recursos do Kubernetes no console do Amazon EKS, conceda a elas Permissões obrigatórias.
- Configure o registro em log para o seu cluster
- Familiarize-se com o que acontece durante desconexões de rede.
- Adicione nós ao seu cluster
- Pense em configurar um plano de backup para seu `etcd`. O Amazon EKS não é compatível com backup e restauração automatizadas do `etcd` para clusters locais. Para obter mais informações, consulte Fazer backup de um cluster do `etcd` na documentação do Kubernetes. As duas principais opções estão usando o `etcdctl` para automatizar a captura de snapshots ou o uso do backup de volume de armazenamento do Amazon EBS.

Conheça as versões de plataforma do Kubernetes e do Amazon EKS para AWS Outposts

As versões da plataforma do cluster local representam os recursos do cluster Amazon EKS no AWS Outposts. As versões incluem os componentes que são executados no ambiente de gerenciamento do Kubernetes e quais sinalizadores do servidor de API do Kubernetes estão habilitados. Incluem também incluem a versão atual do patch do Kubernetes. Cada versão secundária do Kubernetes tem uma ou mais versões de plataforma associadas. As versões de plataforma para diferentes versões secundárias do Kubernetes são independentes. As versões de plataforma para clusters locais e clusters do Amazon EKS na nuvem são independentes.

Quando uma nova versão secundária do Kubernetes está disponível para clusters locais, como `1.28`, a versão inicial da plataforma para essa versão secundária do Kubernetes começa em `eks-local-outposts.1`. Porém, o Amazon EKS lança novas versões de plataforma periodicamente para habilitar novas configurações do ambiente de gerenciamento do Kubernetes e para fornecer correções de segurança.

Quando novas versões da plataforma do cluster local são disponibilizadas para uma versão secundária:

- O número da versão da plataforma é incrementado (`eks-local-outposts.n+1`).
- O Amazon EKS atualiza automaticamente todos os clusters locais existentes para a versão mais recente da plataforma disponível para a versão secundária do Kubernetes correspondente. Atualizações automáticas de versões de plataforma existentes do são implementadas de forma incremental. O processo de implantação pode levar algum tempo. Caso precise imediatamente dos recursos da versão mais recente da plataforma, você deverá criar um novo cluster local.
- O Amazon EKS pode publicar uma nova AMI do nó com uma versão de patch correspondente. Todas as versões de patch são compatíveis entre o ambiente de gerenciamento do Kubernetes e as AMIs dos nós para uma única versão secundária do Kubernetes.

Novas versões de plataforma não apresentam alterações que podem causar interrupções nem causam interrupções de serviço.

Os clusters locais sempre são criados com a versão da plataforma mais recente disponível (`eks-local-outposts.n`) para a versão especificada do Kubernetes.

As versões recentes e atuais da plataforma estão descritas nas tabelas a seguir.

Kubernetes versão **`1.28`**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.28`: `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `DefaultStorageClass`, `DefaultTolerationSeconds`, `ExtendedResourceToleration`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `PersistentVolumeClaimResize`, `Priority`, `PodSecurity`, `ResourceQuota`, `RuntimeClass`, `ServiceAccount`, `StorageObjectInUseProtection`, `TaintNodesByCondition`, `ValidatingAdmissionPolicy` e `ValidatingAdmissionWebhook`.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
1.28.6	eks-local-outposts.5	Versão atualizada do Bottlerocket para v1.19.3 contendo as correções de erros mais recentes para oferecer suporte à inicialização local no Outposts.	18 de abril de 2024
1.28.6	eks-local-outposts.4	Nova versão da plataforma com correções de segurança e melhorias. Suporte restaurado ou inicialização local no Outposts. Versão Bottlerocket rebaixada para v1.15.1 por questões de compatibilidade.	2 de abril de 2024
1.28.6	eks-local-outposts.3	Nova versão da plataforma com correções de segurança e melhorias.	22 de março de 2024
1.28.6	eks-local-outposts.2	Nova versão da plataforma com correções de segurança e melhorias, kube-proxy atualizado para v1.28.6. AWS IAM Authenticator atualizado para v0.6.17. Plug-in CNI da Amazon VPC para Kubernetes foi rebaixado para v1.13.2 por questões de compatibilidade. Versão do Bottlerocket atualizada para v1.19.2.	8 de março de 2024
1.28.1	eks-local-outposts.1	Lançamento inicial do Kubernetes versão v1.28 para	4 de outubro de 2023

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
		clusters locais do Amazon EKS no Outposts	

Kubernetes versão **1.27**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.27: CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, Priority, PodSecurity, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionPolicy e ValidatingAdmissionWebhook.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
1.27.10	eks-local-outposts.5	Nova plataforma com correções de segurança e melhorias.	2 de abril de 2024
1.27.10	eks-local-outposts.4	Nova plataforma com correções de segurança e melhorias , kube-proxy atualizado para v1.27.10. AWS IAM Authenticator atualizado para v0.6.17. Versão do Bottlerocket atualizada para v1.19.2.	22 de março de 2024
1.27.3	eks-local-outposts.3	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.27.3. Plug-in da CNI da Amazon VPC	14 de julho de 2023

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
		para Kubernetes atualizada para `v1.13.2`.	
`1.27.1`	`eks-local-outposts.2`	Imagem do CoreDNS atualizada para `v1.10.1`	22 de junho de 2023
`1.27.1`	`eks-local-outposts.1`	Lançamento inicial do Kubernetes versão `1.27` para clusters locais do Amazon EKS no Outposts	30 de maio de 2023

Kubernetes versão **`1.26`**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.26`: `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `DefaultStorageClass`, `DefaultTolerationSeconds`, `ExtendedResourceToleration`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `PersistentVolumeClaimResize`, `Priority`, `PodSecurity`, `ResourceQuota`, `RuntimeClass`, `ServiceAccount`, `StorageObjectInUseProtection`, `TaintNodesByCondition`, `ValidatingAdmissionPolicy` e `ValidatingAdmissionWebhook`.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
`1.26.13`	`eks-local-outposts.5`	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para `v1.26.13`. AWS IAM Authenticator atualizado para `v0.6.17`. Versão do Bottlerocket atualizada para `v1.19.2`.	22 de março de 2024

Kubernetes versão **1.25**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.25`: `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `DefaultStorageClass`, `DefaultTolerationSeconds`, `ExtendedResourceToleration`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `PersistentVolumeClaimResize`, `Priority`, `PodSecurity`, `ResourceQuota`, `RuntimeClass`, `ServiceAccount`, `StorageObjectInUseProtection`, `TaintNodesByCondition` e `ValidatingAdmissionWebhook`.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
`1.25.16`	`eks-local-outposts.7`	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para `v1.25.16`. AWS IAM Authenticator atualizado para `v0.6.17`. Versão do Bottlerocket atualizada para `v1.19.2`.	22 de março de 2024
`1.25.11`	`eks-local-outposts.6`	Nova versão da plataforma com correções de segurança e melhorias. `kube-proxy` atualizado para `v1.25.11`. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para `v1.13.2`.	14 de julho de 2023
`1.25.9`	`eks-local-outposts.5`	Nova versão da plataforma com correções de segurança e melhorias.	13 de julho de 2023
`1.25.6`	`eks-local-outposts.4`	Versão do Bottlerocket atualizada para `1.13.2`	2 de maio de 2023

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
1.25.6	eks-local-outposts.3	Sistema operacional da instância do ambiente de gerenciamento do Amazon EKS atualizado para Bottlerocket versão v1.13.1 e plug-in da CNI da Amazon VPC atualizado para a versão v1.12.6.	14 de abril de 2023
1.25.6	eks-local-outposts.2	Aprimoramento da coleta de diagnósticos para instâncias do ambiente de gerenciamento do Kubernetes.	8 de março de 2023
1.25.6	eks-local-outposts.1	Lançamento inicial do Kubernetes versão 1.25 para clusters locais do Amazon EKS no Outposts	1.º de março de 2023

Kubernetes versão **1.24**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.24: DefaultStorageClass, DefaultTolerationSeconds, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, ResourceQuota, ServiceAccount, ValidatingAdmissionWebhook, PodSecurityPolicy, TaintNodesByCondition, StorageObjectInUseProtection, PersistentVolumeClaimResize, ExtendedResourceToleration, CertificateApproval, PodPriority, CertificateSigning, CertificateSubjectRestriction, RuntimeClass e DefaultIngressClass.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
1.24.17	eks-local-outposts.7	Nova versão da plataforma com correções de segurança	22 de março de 2024

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
		e melhorias. kube-proxy atualizado para v1.25.16. AWS IAM Authenticator atualizado para v0.6.17. Versão do Bottlerocket atualizada para v1.19.2.	
1.24.15	eks-local-outposts.6	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.24.15. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para v1.13.2.	14 de julho de 2023
1.24.13	eks-local-outposts.5	Nova versão da plataforma com correções de segurança e melhorias.	13 de julho de 2023
1.24.9	eks-local-outposts.4	Versão do Bottlerocket atualizada para 1.13.2	2 de maio de 2023
1.24.9	eks-local-outposts.3	Sistema operacional da instância do ambiente de gerenciamento do Amazon EKS atualizado para Bottlerocket versão v1.13.1 e plug-in da CNI da Amazon VPC atualizado para a versão v1.12.6.	14 de abril de 2023
1.24.9	eks-local-outposts.2	Aprimoramento da coleta de diagnósticos para instâncias do ambiente de gerenciamento do Kubernetes.	8 de março de 2023

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
1.24.9	eks-local-outposts.1	Lançamento inicial do Kubernetes versão 1.24 para clusters locais do Amazon EKS no Outposts	17 de janeiro de 2023

Kubernetes versão **1.23**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.23: DefaultStorageClass, DefaultTolerationSeconds, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, ResourceQuota, ServiceAccount, ValidatingAdmissionWebhook, PodSecurityPolicy, TaintNodesByCondition, StorageObjectInUseProtection, PersistentVolumeClaimResize, ExtendedResourceToleration, CertificateApproval, PodPriority, CertificateSigning, CertificateSubjectRestriction, RuntimeClass e DefaultIngressClass.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
1.23.17	eks-local-outposts.6	Nova versão da plataforma com correções de segurança e melhorias.	13 de julho de 2023
1.23.17	eks-local-outposts.5	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.23.17. Versão do Bottlerocket atualizada para v1.14.1.	6 de julho de 2023
1.23.15	eks-local-outposts.4	Sistema operacional da instância do ambiente de gerenciamento do Amazon EKS atualizado para Bottlerocket	14 de abril de 2023

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas de release	Data de lançamento
		versão v1.13.1 e plug-in da CNI da Amazon VPC atualizado para a versão v1.12.6.	
1.23.15	eks-local-outposts.3	Aprimoramento da coleta de diagnósticos para instâncias do ambiente de gerenciamento do Kubernetes.	8 de março de 2023
1.23.15	eks-local-outposts.2	Lançamento inicial do Kubernetes versão 1.23 para clusters locais do Amazon EKS no Outposts	17 de janeiro de 2023

Criar uma VPC e sub-redes para clusters do Amazon EKS no AWS Outposts

Ao criar um cluster local, você especifica uma VPC e pelo menos uma sub-rede privada que é executada no Outposts. Este tópico fornece uma visão geral dos requisitos e considerações sobre a VPC e as sub-redes para o cluster local.

Requisitos e considerações para VPCs

Quando você cria um cluster local, a VPC especificada deve atender aos requisitos e considerações a seguir:

- Certifique-se de que a VPC tenha endereços IP suficientes para o cluster local, para todos os nós e para outros recursos do Kubernetes que você queira criar. Se a VPC que você deseja usar não tiver endereços IP suficientes, tente aumentar a quantidade de endereços IP disponíveis. É possível fazer isso associando blocos de Encaminhamento Entre Domínios Sem Classificação (CIDR) com sua VPC. É possível associar blocos CIDR privados (RFC 1918) e públicos (não RFC 1918) à VPC antes ou depois de criar o cluster. Um cluster pode levar até cinco horas para reconhecer um bloco CIDR que você associou a uma VPC.

- A VPV não pode ter prefixos IP ou blocos CIDR IPv6 atribuídos. Por causa dessas restrições, as informações que são abordadas em Aumente a quantidade de endereços IP disponíveis para seus nós do Amazon EC2 eEndereços IPv6 para clusters, Pods e services não são aplicáveis à VPC.
- A VPC tem um nome de host DNS e a resolução de DNS habilitados. Sem esses recursos, haverá falha na criação do cluster e será necessário habilitar os recursos e recriar o cluster. Para mais informações, consulte Atributos de DNS para sua VPC, no Guia do usuário da Amazon VPC.
- Para acessar o cluster local pela sua rede local, a VPC deve estar associada à tabela de rotas do gateway local do Outpost. Para obter mais informações, consulte Associações de VPC no Guia do usuário do AWS Outposts.

Requisitos e considerações para sub-redes

Quando você criar o cluster, especifique pelo menos uma sub-rede privada. Se você especificar mais de uma sub-rede, o ambiente de gerenciamento das instâncias do Kubernetes serão distribuídas uniformemente pelas sub-redes. Se mais de uma sub-rede for especificada, as sub-redes deverão estar no mesmo Outpost. Além disso, as sub-redes também devem ter rotas adequadas e permissões de grupo de segurança para se comunicarem entre si. As sub-redes que você especificar ao criar um cluster local devem atender aos requisitos a seguir:

- As sub-redes devem estar todas no mesmo Outpost lógico.
- Juntas as sub-redes devem ter pelo menos três endereços IP disponíveis para as instâncias do ambiente de gerenciamento do Kubernetes. Se três sub-redes forem especificadas, cada uma delas deverá ter pelo menos um endereço IP disponível. Se duas sub-redes forem especificadas, cada uma delas deverá ter pelo menos dois endereços IP disponíveis. Se uma sub-rede for especificada, ela deverá ter pelo menos três endereços IP disponíveis.
- As sub-redes têm uma roda para o gateway local do rack do Outpost para acessar o servidor de API do Kubernetes pela rede local. Se as sub-redes não tiverem uma rota para o gateway local do rack do Outpost, você deverá se comunicar com o servidor de API do Kubernetes de dentro da VPC.
- As sub-redes devem utilizar uma nomenclatura baseada em endereço IP. A nomenclatura baseada em recursos do Amazon EC2 não é compatível com o Amazon EKS.

Acesso à sub-rede para Serviços da AWS

As sub-redes privadas do cluster local no Outposts devem ser capazes de se comunicar com os Serviços da AWS regionais. É possível conseguir isso ao usar um gateway NAT para acesso de

saída à Internet ou, caso deseje manter todo o tráfego privado em sua VPC, ao usar endpoints da VPC de interface.

Como usar um gateway NAT

As sub-redes privadas do cluster local no Outposts devem ter uma tabela de rotas associada que tenha uma rota para um gateway NAT em uma sub-rede pública que esteja na zona de disponibilidade principal do Outpost. A sub-rede pública deve ter uma rota para um gateway da Internet. O gateway NAT permite acesso de saída à Internet e impede conexões de entrada não solicitadas da Internet para instâncias no Outpost.

Usar VPC endpoints da interface do

Se as sub-redes privadas do cluster local no Outposts não tiverem uma conexão de saída com a Internet ou se você desejar manter todo o tráfego privado em sua VPC, crie os endpoints da VPC de interface e o endpoint do gateway a seguir em uma sub-rede regional antes de criar seu cluster.

Endpoint	Tipo de endpoint
com.amazonaws. *region-code* .ssm	Interface
com.amazonaws. *region-code* .ssmmessages	Interface
com.amazonaws. *region-code* .ec2messages	Interface
com.amazonaws. *region-code* .ec2	Interface
com.amazonaws. *region-code* .secretsmanager	Interface
com.amazonaws. *region-code* .logs	Interface
com.amazonaws. *region-code* .sts	Interface
com.amazonaws. *region-code* .ecr.api	Interface
com.amazonaws. *region-code* .ecr.dkr	Interface
com.amazonaws. *region-code* .s3	Gateway

Os endpoints devem atender aos seguintes requisitos:

- Ter sido criado em uma sub-rede privada localizada na zona de disponibilidade principal do Outpost.
- Ter nomes DNS privados habilitados.
- Ter um grupo de segurança anexado que permita o tráfego HTTPS de entrada do intervalo CIDR da sub-rede privada do Outpost.

A criação de endpoints gera cobranças. Para obter mais informações, consulte Preços do AWS PrivateLink. Se os Pods precisarem acessar outros Serviços da AWS, você precisará criar endpoints adicionais. Para obter uma lista abrangente de endpoints, consulte Serviços da AWS que se integram ao AWS PrivateLink.

Crie uma VPC

Você pode criar uma VPC que atenda aos requisitos anteriores usando um dos seguintes modelos de AWS CloudFormation:

- Modelo 1: esse modelo cria uma VPC com uma sub-rede privada no Outpost e uma sub-rede pública na Região da AWS. A sub-rede privada tem uma rota para a Internet passando por um gateway NAT que reside na sub-rede pública da Região da AWS. Esse modelo pode ser usado para criar um cluster local em uma sub-rede com acesso de egresso à Internet.
- Modelo 2: esse modelo cria uma VPC com uma sub-rede privada no Outpost e com o conjunto mínimo de endpoints da VPC necessário para criar um cluster local em uma sub-rede que não tem acesso de ingresso e egresso à Internet (também chamada de sub-rede privada).

Preparar clusters locais do Amazon EKS no AWS Outposts para desconexões de rede

Se a rede local tiver perdido a conectividade com a Nuvem AWS, você pode continuar a usar o cluster local do Amazon EKS em um Outpost. Este tópico aborda como você pode preparar o cluster local para desconexões de rede e as considerações relacionadas.

Considerações ao preparar o cluster local para uma desconexão de rede:

- Os clusters locais permitem estabilidade e operações contínuas durante desconexões temporárias e não planejadas da rede. O AWS Outposts continua a ser uma oferta totalmente conectada que

atua como uma extensão da Nuvem AWS no seu data center. No caso de desconexões de rede entre o Outpost e a Nuvem AWS, recomendamos que você tente restaurar a conexão. Para obter instruções, consulte a Lista de verificação da solução de problemas da rede de racks do AWS Outposts no Guia do usuário do AWS Outposts. Para obter informações sobre como solucionar problemas de clusters locais, consulte Solucionar problemas de clusters locais do Amazon EKS no AWS Outposts.

- Os Outposts emitem uma métrica `ConnectedStatus` que você pode usar para monitorar o estado da conectividade do Outpost. Para obter mais informações, consulte Métricas do Outposts no Guia do usuário do AWS Outposts.
- Os clusters locais usam o IAM como mecanismo de autenticação padrão usando o Autenticador do AWS Identity and Access Management para Kubernetes. O IAM não está disponível durante desconexões de rede. Assim, os clusters locais são compatíveis com um mecanismo de autenticação alternativo usando certificados `x.509` que você pode usar para se conectar ao cluster durante desconexões de rede. Para obter informações sobre como obter e usar um certificado `x.509` para o cluster, consulte Autenticar no cluster local durante uma desconexão de rede.
- Caso não você consiga acessar o Route 53 durante desconexões de rede, considere o uso de servidores DNS locais no ambiente on-premises. As instâncias do ambiente de gerenciamento do Kubernetes usam endereços IP estáticos. Você pode configurar os hosts que usa para se conectar ao cluster com o nome do host do endpoint e os endereços IP como uma alternativa ao uso de servidores DNS locais. Para obter mais informações, consulte DNS no Guia do usuário do AWS Outposts.
- Se você esperar aumentos no tráfego das aplicações durante desconexões de rede, pode provisionar capacidade computacional adicional no cluster quando estiver conectado à nuvem. As instâncias do Amazon EC2 estão incluídas no preço do AWS Outposts. Portanto, a execução de instâncias adicionais não afeta o custo de uso da AWS.
- Durante desconexões de rede para permitir operações de criação, atualização e escalação de workloads, as imagens de contêiner da aplicação devem estar acessíveis pela rede local e o cluster deve ter capacidade suficiente. Os clusters locais não hospedam um registro de contêiner para você. Se os Pods foram executados anteriormente nesses nós, as imagens de contêiners serão armazenadas em cache nos nós. Se você normalmente extrair imagens de contêiner da aplicação do Amazon ECR na nuvem, considere executar um cache ou registro local. Um cache ou registro local será útil se você precisar de operações de criação, atualização e escalação para recursos de workload durante desconexões de rede.

- Os clusters locais usam o Amazon EBS como a classe de armazenamento padrão para volumes persistentes e o driver da CSI do Amazon EBS para gerenciar o ciclo de vida dos volumes persistentes do Amazon EBS. Durante desconexões de rede, os Pods que são baseados no Amazon EBS não podem ser criados, atualizados nem escalados. Isso porque essas operações exigem chamadas para a API do Amazon EBS na nuvem. Se você estiver implantando workloads com estado em clusters locais e precisar de operações de criação, atualização ou escalabilidade durante desconexões de rede, considere usar um mecanismo de armazenamento alternativo.
- Os snapshots do Amazon EBS não podem ser criados ou excluídos se o AWS Outposts não puder acessar as APIs relevantes da AWS dentro da região (como as APIs do Amazon EBS ou do Amazon S3).
- Ao integrar o ALB (entrada) com o AWS Certificate Manager (ACM), os certificados são enviados e armazenados na memória da instância de computação do ALB do AWS Outposts. A terminação TLS atual continuará operando no caso de uma desconexão do Região da AWS. As operações de mutação nesse contexto falharão (como novas definições de entrada, novas operações de API de certificados baseados em ACM, escala de computação do ALB ou rotação de certificados). Para obter mais informações, consulte Solução de problemas de renovação de certificado gerenciado no Guia do usuário do AWS Certificate Manager.
- Os logs do ambiente de gerenciamento do Amazon EKS são armazenados em cache localmente nas instâncias do ambiente de gerenciamento do Kubernetes durante desconexões de rede. Após a reconexão, os logs são enviados ao CloudWatch Logs na Região da AWS pai. Você pode usar soluções de parceiros do Prometheus, do Grafana ou do Amazon EKS para monitorar o cluster localmente usando o endpoint de métricas do servidor de API do Kubernetes ou usando o Fluent Bit para logs.
- Se você estiver usando o AWS Load Balancer Controller no Outposts para tráfego das aplicações, os Pods existentes fronteados pelo AWS Load Balancer Controller continuarão a receber tráfego durante desconexões de rede. Novos Pods criados durante desconexões de rede só receberão tráfego quando o Outpost for reconectado à Nuvem AWS. Considere definir a contagem de réplicas das suas aplicações durante a conexão com a Nuvem AWS para acomodar suas necessidades de escalabilidade durante desconexões de rede.
- O Amazon VPC CNI plugin for Kubernetes assume o modo IP secundário por padrão. Ele é configurado com `WARM_ENI_TARGET = 1`, o que permite que o plug-in mantenha disponível "uma interface de rede totalmente elástica" de endereços IP. Considere mudar os valores `WARM_ENI_TARGET`,`WARM_IP_TARGET` e `MINIMUM_IP_TARGET` de acordo com suas necessidades de escalonamento durante um estado desconectado. Para obter mais informações ,

consulte o arquivo `readme` no GitHub. Para obter uma lista do número máximo de Pods compatível com cada tipo de instância, consulte o arquivo `eni-max-pods.txt` no GitHub.

Autenticar no cluster local durante uma desconexão de rede

O AWS Identity and Access Management (IAM) não está disponível durante desconexões de rede. Você não pode autenticar no cluster local usando credenciais do IAM enquanto está desconectado. Porém, você pode se conectar ao cluster pela rede local usando certificados `x509` quando estiver desconectado. Você precisa baixar e armazenar o certificado `X509` de um cliente para usar durante as desconexões. Neste tópico, você aprenderá a criar e usar o certificado para autenticação no cluster quando ele está em um estado desconectado.

1. Crie uma solicitação de assinatura do certificado.

   a. Gere uma solicitação de assinatura do certificado.

   ```
   openssl req -new -newkey rsa:4096 -nodes -days 365 \
       -keyout admin.key -out admin.csr -subj "/CN=admin"
   ```

   b. Crie uma solicitação de assinatura do certificado no Kubernetes.

   ```
   BASE64_CSR=$(cat admin.csr | base64 -w 0)
   cat << EOF > admin-csr.yaml
   apiVersion: certificates.k8s.io/v1
   kind: CertificateSigningRequest
   metadata:
     name: admin-csr
   spec:
     signerName: kubernetes.io/kube-apiserver-client
     request: ${BASE64_CSR}
     usages:
     - client auth
   EOF
   ```

2. Crie uma solicitação de assinatura do certificado usando o `kubectl`.

   ```
   kubectl create -f admin-csr.yaml
   ```

3. Verifique o status da solicitação de assinatura do certificado.

```
kubectl get csr admin-csr
```

Veja um exemplo de saída abaixo.

```
NAME       AGE   REQUESTOR                      CONDITION
admin-csr  11m   kubernetes-admin               Pending
```

O Kubernetes criou a solicitação de assinatura do certificado.

4. Aprove a solicitação de assinatura do certificado.

```
kubectl certificate approve admin-csr
```

5. Verifique novamente o status da solicitação de assinatura do certificado para aprovação.

```
kubectl get csr admin-csr
```

Veja um exemplo de saída abaixo.

```
NAME       AGE   REQUESTOR                    CONDITION
admin-csr  11m   kubernetes-admin             Approved
```

6. Recupere e verifique o certificado.

a. Recupere o certificado.

```
kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode
 > admin.crt
```

b. Verifique o certificado.

```
cat admin.crt
```

7. Crie uma vinculação de função de cluster para um usuário `admin`.

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
    --user=admin --group=system:masters
```

8. Gere um kubeconfig com escopo de usuário para um estado desconectado.

Você pode gerar um arquivo `kubeconfig` usando os certificados `admin` baixados. Substitua *`my-cluster`* e *`apiserver-endpoint`* nos comandos a seguir.

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.certificateAuthority" \
    --output text | base64 --decode > ca.crt
```

```
kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
    --certificate-authority=ca.crt --server apiserver-endpoint --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
    --client-certificate=admin.crt --client-key=admin.key --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
    --cluster my-cluster --user admin
```

```
kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
```

9. Visualize o arquivo `kubeconfig`.

```
kubectl get nodes --kubeconfig admin.kubeconfig
```

10. Se você já tiver serviços em produção no Outpost, pule esta etapa. Se o Amazon EKS for o único serviço em execução no Outpost e o Outpost não estiver em produção no momento, você poderá simular uma desconexão de rede. Antes de entrar em produção com o cluster local, simule uma desconexão para ter certeza de que pode acessar o cluster mesmo quando ele está no estado desconectado.

    a. Aplique regras de firewall nos dispositivos de rede que conectam o Outpost à Região da AWS. Com isso, o link de serviço do Outpost é desconectado. Você não pode criar novas instâncias. As instâncias atualmente em execução perdem conectividade com a Região da AWS e a Internet.

    b. Você pode testar a conexão com o cluster local enquanto estiver desconectado usando o certificado `x509`. Certifique-se de alterar o `kubeconfig` para o `admin.kubeconfig` que você criou em uma etapa anterior. Substitua *`my-cluster`* pelo nome do cluster local.

```
kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
```

Se você notar algum problema com os clusters locais enquanto eles estiverem no estado desconectado, será recomendável abrir um tíquete de suporte.

Selecionar tipos de instância e grupos de posicionamento para clusters do Amazon EKS no AWS Outposts com base em considerações de capacidade

Este tópico fornece orientação para a seleção de tipo de instância do ambiente de gerenciamento do Kubernetes e (opcionalmente) o uso de grupos de posicionamento para atender aos requisitos de alta disponibilidade do cluster local do Amazon EKS em um Outpost.

Antes de selecionar um tipo de instância (como `m5`, `c5` ou `r5`) para usar no ambiente de gerenciamento do Kubernetes do cluster local no Outposts, confirme os tipos de instâncias que estão disponíveis na configuração do Outpost. Depois que você identificar os tipos de instância disponíveis, selecione o tamanho da instância (por exemplo, `large`, `xlarge` ou `2xlarge`) com base no número de nós que as workloads exigem. A tabela a seguir fornece recomendações para a escolha do tamanho de uma instância.

> **Note**
>
> Os tamanhos das instâncias devem ser alocados nos Outposts. Certifique-se de ter capacidade suficiente para três instâncias do tamanho disponível no Outposts durante a vida útil do cluster local. Para obter uma lista dos tipos de instância do Amazon EC2 disponíveis, consulte as seções Computação e Armazenamento em Recursos de rack do AWS Outposts.

Número de nós	Tamanho da instância do ambiente de gerenciamento do Kubernetes
1–20	`large`
21–100	`xlarge`

Número de nós	Tamanho da instância do ambiente de gerenciamento do Kubernetes
101–250	`2xlarge`
251–500	`4xlarge`

O armazenamento para o ambiente de gerenciamento do Kubernetes requer 246 GB do armazenamento do Amazon EBS para cada cluster local para atender aos requisitos de IOPS do `etcd`. Na criação do cluster local, os volumes do Amazon EBS são provisionados automaticamente.

Posicionamento do ambiente de gerenciamento

Quando você não especifica um grupo de posicionamento com a propriedade `OutpostConfig.ControlPlanePlacement.GroupName`, as instâncias do Amazon EC2 provisionadas para o ambiente de gerenciamento do Kubernetes não recebem nenhuma imposição específica de posicionamento de hardware em toda a capacidade subjacente disponível no Outpost.

Você pode usar grupos de posicionamento para atender aos requisitos de alta disponibilidade do cluster local do Amazon EKS em um Outpost. Especificando um grupo de posicionamento durante a criação do cluster, você influencia o posicionamento das instâncias do ambiente de gerenciamento do Kubernetes. As instâncias são distribuídas pelo hardware subjacente independente (racks ou hosts), minimizando o impacto das instância correlacionadas no caso de falhas de hardware.

Requisitos

O tipo de distribuição que você pode configurar depende do número de racks do Outpost que você tem na implantação.

- Implantações com um ou dois racks físicos em um único Outpost lógico: é necessário ter pelo menos três hosts configurados com o tipo de instância que você escolher para as suas instâncias do ambiente de gerenciamento do Kubernetes. Um grupo de posicionamento distribuído usando a distribuição no nível dos hosts garante que todas as instâncias do ambiente de gerenciamento do Kubernetes sejam executadas em hosts distintos, em todos os racks subjacentes disponíveis na implantação do Outpost.
- Implantações com três ou mais racks físicos em um único Outpost lógico: é necessário ter, pelo menos, três hosts configurados com o tipo de instância que você escolher para as suas instâncias do ambiente de gerenciamento do Kubernetes. Um grupo de posicionamento distribuído usando a

distribuição no nível dos racks garante que todas as instâncias do ambiente de gerenciamento do Kubernetes sejam executadas em racks distintos na implantação do Outpost. Ou então, você pode usar o grupo de posicionamento de distribuição no nível dos hosts, conforme descrito na opção anterior.

Você é responsável por criar o grupo de posicionamento desejado. Você especifica o grupo de posicionamento ao chamar a API `CreateCluster`. Para obter mais informações sobre grupos de posicionamento e como criá-los, consulte Grupos de posicionamento no Guia do usuário do Amazon EC2.

Considerações

- Quando um grupo de posicionamento é especificado, deve haver capacidade disponível alocada no Outpost para a criação bem-sucedida de um cluster local do Amazon EKS. A capacidade varia dependendo do tipo de distribuição que você usa: distribuição em hosts ou em racks. Se não houver capacidade suficiente, o cluster permanecerá no estado `Creating`. Você pode verificar o `Insufficient Capacity Error` no campo de integridade da resposta da API `DescribeCluster`. Você deve liberar capacidade para que o processo de criação prossiga.
- Durante as atualizações de versão e de plataforma do cluster local do Amazon EKS, as instâncias do ambiente de gerenciamento do Kubernetes do cluster são substituídas por novas instâncias usando uma estratégia de atualização contínua. Durante esse processo de substituição, cada instância do ambiente de gerenciamento é encerrada, liberando o slot correspondente. Uma nova instância atualizada é provisionada em seu lugar. A instância atualizada pode ser posicionada no slot que foi liberado. Se o slot for consumido por outra instância não relacionada e não restar mais capacidade para atender ao requisito de topologia de distribuição, o cluster permanecerá no estado `Updating`. Você pode ver o `Insufficient Capacity Error` correspondente no campo de integridade da resposta da API `DescribeCluster`. Você deve liberar capacidade para que o processo de atualização possa prosseguir e restabelecer os altos níveis de disponibilidade anteriores.
- É possível criar até 500 grupos de posicionamento por conta em cada Região da AWS. Para obter mais informações, consulte Regras gerais e limitações no Guia do usuário do Amazon EC2.

Solucionar problemas de clusters locais do Amazon EKS no AWS Outposts

Este tópico aborda alguns erros comuns que você pode encontrar ao usar clusters locais e como solucionar esses problemas. Os clusters locais são semelhantes aos clusters do Amazon EKS na nuvem, mas existem algumas diferenças na forma como eles são gerenciados pelo Amazon EKS.

Comportamento da API

Os clusters locais são criados por meio da API do Amazon EKS, mas são executados de maneira assíncrona. Isso significa que as solicitações à API do Amazon EKS retornam imediatamente para os clusters locais. Porém, essas solicitações podem ser bem-sucedidas, antecipar-se à falha devido a erros de validação de entrada ou falhar e ter erros de validação descritivos. Esse comportamento é semelhante ao da API do Kubernetes.

Os clusters locais não fazem a transição para um status `FAILED`. O Amazon EKS tenta continuamente reconciliar o estado do cluster com o estado desejado solicitado pelo usuário. Como resultado, um cluster local pode permanecer no estado `CREATING` por um período prolongado até que o problema subjacente seja resolvido.

Descrever o campo de integridade do cluster

Problemas de cluster local podem ser descobertos usando o comando `describe-cluster` da AWS CLI do Amazon EKS. Problemas de cluster local são revelados pelo campo `cluster.health` da resposta do comando `describe-cluster`. A mensagem contida nesse campo inclui um código de erro, uma mensagem descritiva e IDs de recursos relacionados. Essas informações só estão disponíveis por meio da API do Amazon EKS e da AWS CLI. No exemplo a seguir, substitua *my-cluster* pelo nome do cluster local.

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

Veja um exemplo de saída abaixo.

```
{
    "issues": [
        {
            "code": "ConfigurationConflict",
            "message": "The instance type 'm5.large' is not supported in Outpost 'my-
outpost-arn'.",
            "resourceIds": [
                "my-cluster-arn"
```

```
            ]
        }
    ]
}
```

Se o problema não puder ser reparado, talvez seja necessário excluir o cluster local e criar um novo. Por exemplo, tentar provisionar um cluster com um tipo de instância que não está disponível no Outpost. A tabela a seguir inclui erros comuns relacionados à integridade.

Cenário de erro	Código	Message	ResourceIds
Não foi possível encontrar as sub-redes fornecidas.	`ResourceNotFound`	`The subnet ID` *`subnet-id`* `does not exist`	Todos os IDs de sub-rede fornecidos
As sub-redes fornecidas não pertencem à mesma VPC.	`ConfigurationConflict`	`Subnets specified must belong to the same VPC`	Todos os IDs de sub-rede fornecidos
Algumas sub-redes fornecidas não pertencem ao Outpost especificado.	`ConfigurationConflict`	`Subnet` *`subnet-id`* `expected to be in` *`outpost-arn`* `, but is in` *`other-outpost-arn`*	ID de sub-rede problemática
Algumas sub-redes fornecidas não pertencem a nenhum Outpost.	`ConfigurationConflict`	`Subnet` *`subnet-id`* `is not part of any Outpost`	ID de sub-rede problemática
Algumas sub-redes fornecidas não têm endereços livres suficientes para a criação de interfaces de rede elásticas para as instância	`ResourceLimitExceeded`	`The specified subnet does not have enough free addresses to satisfy the request.`	ID de sub-rede problemática

Cenário de erro	Código	Message	ResourceIds
s do ambiente de gerenciamento.			
O tipo de instância do ambiente de gerenciamento especificado não é compatível com o Outpost.	`ConfigurationConflict`	`The instance type` *`type`* `is not supported in Outpost` *`outpost-arn`*	ARN do cluster
Você encerrou uma instância do Amazon EC2 do ambiente de gerenciamento ou `run-instance` teve êxito, mas o estado sofreu alterações para `Terminated`. Isso pode acontecer por um período após a reconexão do Outpost e os erros internos do Amazon EBS causarem uma falha no fluxo de trabalho interno do Amazon EC2.	`InternalFailure`	`EC2 instance state "Terminated" is unexpected`	ARN do cluster

Cenário de erro	Código	Message	ResourceIds
Você não tem capacidade suficiente no Outpost. Isso também pode acontecer durante a criação do cluster se um Outpost estiver desconectado da Região da AWS.	`ResourceLimitExceeded`	`There is not enough capacity on the Outpost to launch or start the instance.`	ARN do cluster
A conta excedeu a cota de grupo de segurança.	`ResourceLimitExceeded`	Mensagem de erro retornada pela API do Amazon EC2	ID da VPC de destino
A conta excedeu a cota de interface de rede elástica.	`ResourceLimitExceeded`	Mensagem de erro retornada pela API do Amazon EC2	ID da sub-rede de destino
As instâncias do ambiente de gerenciamento não podem ser acessadas por meio do AWS Systems Manager. Para saber a resolução, consulte Instâncias do ambiente de gerenciamento não podem ser acessadas por meio do AWS Systems Manager.	`ClusterUnreachable`	As instâncias do ambiente de gerenciamento do Amazon EKS não podem ser acessadas por meio do SSM. Verifique a configuração do SSM e da rede e consulte a documentação de solução de problemas do EKS no Outposts.	IDs de instâncias do Amazon EC2

Cenário de erro	Código	Message	ResourceIds
Ocorreu um erro ao obter detalhes de um grupo de segurança gerenciado ou de uma interface de rede elástica.	Com base no código de erro do cliente Amazon EC2.	Mensagem de erro retornada pela API do Amazon EC2	Todos os IDs de grupos de segurança gerenciados
Ocorreu um erro ao autorizar ou revogar as regras de ingresso de grupo de segurança. Isso se aplica aos grupos de segurança do cluster e do ambiente de gerenciamento.	Com base no código de erro do cliente Amazon EC2.	Mensagem de erro retornada pela API do Amazon EC2	ID de grupo de segurança problemát ico
Ocorreu um erro ao excluir uma interface de rede elástica de uma instância do ambiente de gerenciamento.	Com base no código de erro do cliente Amazon EC2.	Mensagem de erro retornada pela API do Amazon EC2	ID de interface de rede elástica problemática

A tabela a seguir lista os erros de outros Serviços da AWS que são apresentados no campo de integridade da resposta de `describe-cluster`.

Código de erro do Amazon EC2	Código de problema de integridade do cluster	Descrição
`AuthFailure`	`AccessDenied`	Esse erro pode ocorrer por vários motivos. O motivo mais comum é que você acidental mente removeu do serviço do ambiente de gerenciam

Código de erro do Amazon EC2	Código de problema de integridade do cluster	Descrição
		ento uma tag que o serviço usa para reduzir o escopo da política de perfil vinculada. Se isso ocorrer, o Amazon EKS não poderá mais gerenciar e monitorar esses recursos da AWS.
`UnauthorizedOperation`	`AccessDenied`	Esse erro pode ocorrer por vários motivos. O motivo mais comum é que você acidentalmente removeu do serviço do ambiente de gerenciamento uma tag que o serviço usa para reduzir o escopo da política de perfil vinculada. Se isso ocorrer, o Amazon EKS não poderá mais gerenciar e monitorar esses recursos da AWS.
`InvalidSubnetID.NotFound`	`ResourceNotFound`	Esse erro ocorre quando o ID da sub-rede para as regras de ingresso de um grupo de segurança não pode ser encontrado.
`InvalidPermission.NotFound`	`ResourceNotFound`	Esse erro ocorre quando as permissões para as regras de ingresso de um grupo de segurança não estão corretas.

Código de erro do Amazon EC2	Código de problema de integridade do cluster	Descrição
`InvalidGroup.NotFound`	`ResourceNotFound`	Esse erro ocorre quando o grupo das regras de ingresso de um grupo de segurança não pode ser encontrado.
`InvalidNetworkInterfaceID.NotFound`	`ResourceNotFound`	Esse erro ocorre quando o ID da interface de rede para as regras de ingresso de um grupo de segurança não pode ser encontrado.
`InsufficientFreeAddressesInSubnet`	`ResourceLimitExceeded`	Esse erro ocorre quando a cota de recursos da sub-rede é excedida.
`InsufficientCapacityOnOutpost`	`ResourceLimitExceeded`	Esse erro ocorre quando a cota de capacidade do outpost é excedida.
`NetworkInterfaceLimitExceeded`	`ResourceLimitExceeded`	Esse erro ocorre quando a cota de interface de rede elástica é excedida.
`SecurityGroupLimitExceeded`	`ResourceLimitExceeded`	Esse erro ocorre quando a cota de capacidade do grupo de segurança é excedida.

Código de erro do Amazon EC2	Código de problema de integridade do cluster	Descrição
`VcpuLimitExceeded`	`ResourceLimitExceeded`	Isso é observado na criação de uma instância do Amazon EC2 em uma nova conta. O procedimento pode ser semelhante ao seguinte: `"You have requested more vCPU capacity than your current vCPU limit of 32 allows for the instance bucket that the specified instance type belongs to. Please visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to this limit."`
`InvalidParameterValue`	`ConfigurationConflict`	O Amazon EC2 retornará esse código de erro se não houver suporte no Outpost para o tipo de instância especificado.
Todas as outras falhas	`InternalFailure`	Nenhum

Impossível criar ou modificar clusters

Os clusters locais exigem permissões e políticas diferentes das exigidas pelos clusters do Amazon EKS hospedados na nuvem. Quando a criação do cluster falhar e gera um erro `InvalidPermissions`, verifique novamente se o perfil do cluster que você está usando tem a política gerenciada AmazonEKSLocalOutpostClusterPolicy anexada a ele. Todas as outras chamadas de API exigem o mesmo conjunto de permissões que os clusters do Amazon EKS na nuvem.

O cluster está travado no estado **`CREATING`**

O tempo necessário para criar um cluster local varia dependendo de vários fatores. Esses fatores incluem a configuração da rede, a configuração do Outpost e a configuração do cluster. Em geral, um cluster local é criado e passa para o status `ACTIVE` dentro de 15 a 20 minutos. Se um cluster local permanecer no estado `CREATING`, você poderá chamar `describe-cluster` para obter informações sobre a causa no campo de saída `cluster.health`.

Os problemas mais comuns são os seguintes:

O AWS Systems Manager (Systems Manager) encontra os seguintes problemas:

- O cluster não pode se conectar à instância do ambiente de gerenciamento na Região da AWS em que o Systems Manager está. Você pode verificar isso chamando `aws ssm start-session --target` *`instance-id`* em um bastion host na região. Se esse comando não funcionar, verifique se o Systems Manager está sendo executado na instância do ambiente de gerenciamento. Outra solução de contorno é excluir o cluster e depois recriá-lo.
- As instâncias do ambiente de gerenciamento do Systems Manager podem não ter acesso à Internet. Verifique se a sub-rede que você forneceu ao criar o cluster tem um gateway NAT e uma VPC com um gateway da internet. Use o analisador de acessibilidade da VPC para verificar se a instância do ambiente de gerenciamento pode acessar o gateway da Internet. Para obter mais informações, consulte Getting started with VPC Reachability Analyzer (Conceitos básicos do VPC Reachability Analyzer).
- O ARN de perfil fornecido não tem políticas. Verifique se a Política gerenciada pela AWS: AmazonEKSLocalOutpostClusterPolicy foi removida do perfil. Isso também pode ocorrer se uma pilha AWS CloudFormation estiver configurada incorretamente.

Várias sub-redes são mal configuradas e especificadas quando um cluster é criado:

- Todas as sub-redes fornecidas devem estar associadas ao mesmo Outpost e devem alcançar umas às outras. Quando várias sub-redes são especificadas durante a criação do cluster, o Amazon EKS tenta distribuir as instâncias do ambiente de gerenciamento entre várias sub-redes.
- Os grupos de segurança gerenciados pelo Amazon EKS são aplicados à interface de rede elástica. No entanto, outros elementos da configuração, como regras de firewall NACL, podem entrar em conflito com as regras da interface de rede elástica.

A VPC e o DNS da sub-rede estão mal configurados ou falta a configuração

Consulte Criar uma VPC e sub-redes para clusters do Amazon EKS no AWS Outposts.

Não é possível integrar nós a um cluster

Causas comuns:

- Problemas de AMI:
  - Você está usando uma AMI incompatível. Você deve estar usando a v20220620 ou posterior do Amazon Linux otimizado pelo Amazon EKS da Criar nós com AMIs do Amazon Linux otimizadas.
  - Se você usou um modelo do AWS CloudFormation para criar os nós, verifique se ele não estava usando uma AMI incompatível.
- O `ConfigMap` do autenticador do AWS IAM está ausente: nesse caso, é necessário criá-lo. Para obter mais informações, consulteComo aplicar o `ConfigMapaws-auth` ao seu cluster
- O grupo de segurança errado é usado: certifique-se de usar `eks-cluster-sg-`*`cluster-name`*`-`*`uniqueid`* para o grupo de segurança dos nós de processamento. O grupo de segurança selecionado é alterado pelo AWS CloudFormation para permitir um novo grupo de segurança sempre que a pilha for usada.
- Seguir etapas inesperadas de uma VPC de link privado: são especificados dados de CA (`--b64-cluster-ca`) ou endpoint de API (`--apiserver-endpoint`) incorretos.
- Política de segurança de Pod mal configurada:
  - Os daemonsets CoreDNS e Amazon VPC CNI plugin for Kubernetes devem ser executados nos nós para que eles se integrem e se comuniquem com o cluster.
  - O Amazon VPC CNI plugin for Kubernetes requer alguns recursos de rede privilegiada para funcionar corretamente. É possível visualizar os recursos de rede privilegiada com o seguinte comando: `kubectl describe psp eks.privileged`.

  Recomendamos que você não modifique a política padrão de segurança de pods. Para obter mais informações, consulte Política de segurança de pods.

Coletar logs

Quando um Outpost é desconectado da Região da AWS à qual está associado, o cluster do Kubernetes provavelmente continuará funcionando normalmente. Mas, se o cluster não funcionar corretamente, siga as etapas de solução de problemas em Preparar clusters locais do Amazon EKS no AWS Outposts para desconexões de rede. Se você encontrar outros problemas, entre em contato com o AWS Support. O AWS Support pode orientá-lo no download e na execução de

uma ferramenta de coleta de logs. Assim, você poderá coletar logs das instâncias do ambiente de gerenciamento do cluster do Kubernetes e enviá-las ao AWS Support para uma investigação adicional.

Instâncias do ambiente de gerenciamento não podem ser acessadas por meio do AWS Systems Manager

Quando as instâncias do ambiente de gerenciamento do Amazon EKS não são acessíveis por meio do AWS Systems Manager (Systems Manager), o Amazon EKS exibe o erro para o cluster.

```
Amazon EKS control plane instances are not reachable through SSM. Please verify your
 SSM and network configuration, and reference the EKS on Outposts troubleshooting
 documentation.
```

Para solucionar esse problema, certifique-se de que a VPC e as sub-redes atendam aos requisitos em Criar uma VPC e sub-redes para clusters do Amazon EKS no AWS Outposts e que você concluiu as etapas em Configurar o Session Manager no Guia do usuário do AWS Systems Manager.

Criar nós Amazon Linux no AWS Outposts

Este tópico descreve como iniciar um grupo do Auto Scaling de nós do Amazon Linux em um Outpost que são registrados no cluster do Amazon EKS. O cluster pode estar na Nuvem AWS ou em um Outpost.

Pré-requisitos

- Um Outpost existente. Para obter mais informações, consulte O que é o AWS Outposts.
- Um cluster do Amazon EKS existente. Para implantar um cluster na Nuvem AWS, consulte Criar um cluster do Amazon EKS.. Para implantar um cluster em um Outpost, consulte Criar clusters locais do Amazon EKS no AWS Outposts para garantir alta disponibilidade.
- Suponha que você esteja criando os nós em um cluster na Nuvem AWS e que tenha sub-redes na Região da AWS em que o AWS Outposts, o AWS Wavelength ou as zonas locais da AWS estão habilitados. Assim sendo, essas sub-redes não devem ter sido transmitidas quando você criou o cluster. Se você estiver criando nós em um cluster em um Outpost, deverá ter transmitido uma sub-rede do Outpost ao criar o cluster.
- (Recomendado para clusters na Nuvem AWS) O complemento Amazon VPC CNI plugin for Kubernetes configurado com seu o próprio perfil do IAM tem a política do IAM necessária anexada a ele. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for

Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA). Clusters locais não são compatíveis com perfis do IAM para contas de serviço.

Você pode criar um grupo de nós autogerenciado do Amazon Linux com o `eksctl` ou o AWS Management Console (com um modelo do AWS CloudFormation). Você também pode usar o Terraforma.

eksctl

Pré-requisito

Versão `0.187.0` ou posterior da ferramenta da linha de comando do `eksctl` instalada no dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Para iniciar nós autogerenciados do Linux usando **`eksctl`**

1. Se o cluster estiver na Nuvem AWS e a política gerenciada do IAM AmazonEKS_CNI_Policy estiver anexada ao Perfil do IAM em nós do Amazon EKS, é recomendável atribuí-la a um perfil do IAM que você associa à conta de serviço `aws-node` do Kubernetes. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA). Se o cluster estiver no Outpost, a política deverá estar anexada à função do nó.
2. O comando a seguir cria um grupo de nós em um cluster existente. O cluster deve ter sido criado usando `eksctl`. Substitua *`al-nodes`* por um nome para o seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Substitua o *`my-cluster`* pelo nome do cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster. Se o cluster estiver em um Outpost, substitua *`id`* pelo ID de uma sub-rede do Outpost. Se o cluster estiver na Nuvem AWS, substitua *`id`* pelo ID de uma sub-rede que você não especificou ao criar o cluster. Substitua *`instance-type`* por um tipo de instância com suporte pelo seu Outpost. Substitua os valores de *`example values`* restantes pelos seus próprios valores. Os nós são criados com a mesma versão do Kubernetes que o ambiente de gerenciamento, por padrão.

   Substitua *`instance-type`* por um tipo de instância disponível em seu Outpost.

Substituir *my-key* pelo nome do par de chaves do Amazon EC2 ou da chave pública. Essa chave é usada para SSH em seus nós depois que eles forem iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no AWS Management Console. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2.

Crie seu grupo de nós com o comando a seguir.

```
eksctl create nodegroup --cluster my-cluster --name al-nodes --node-
type instance-type \
    --nodes 3 --nodes-min 1 --nodes-max 4 --managed=false --node-volume-type gp2
 --subnet-ids subnet-id
```

Se o cluster for implantado na Nuvem AWS:

- O grupo de nós que você implantar poderá atribuir endereços `IPv4` aos Pods de um bloco CIDR diferente do bloco da instância. Para obter mais informações, consulte Rede personalizada para pods.
- O grupo de nós que você implantar não exigirá acesso de saída à Internet. Para obter mais informações, consulte Implementar clusters privados com acesso limitado à internet.

Para obter uma lista completa de todas as opções e padrões disponíveis, consulte Suporte do AWS Outposts na documentação do `eksctl`.

Se os nós não se integrarem ao cluster, consulte Worker nodes fail to join cluster (Falha nos nós ao ingressar no cluster) em Solucionar problemas com clusters e nós do Amazon EKS e Não é possível integrar nós a um cluster em Solucionar problemas de clusters locais do Amazon EKS no AWS Outposts.

Veja um exemplo de saída abaixo. Várias linhas são emitidas enquanto os nós são criados. Uma das últimas linha de saída é o exemplo de linha a seguir.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opcional) Implante uma aplicação de amostra para testar o cluster e os nós do Linux.

AWS Management Console

Etapa 1: Para iniciar nós autogerenciados do Amazon Linux usando o AWS Management Console

1. Faça download da versão mais recente do modelo AWS CloudFormation.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

2. Abra o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation.
3. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos, padrão).
4. Para Specify template (Especificar modelo), selecione Upload a template file (Fazer upload de um arquivo de modelo) e depois Choose file (Escolher arquivo). Selecione o arquivo `amazon-eks-nodegroup.yaml` que você baixou em uma etapa anterior e selecione Next (Próximo).
5. Na página Specify stack details (Especificar detalhes da pilha), insira os parâmetros de acordo e escolha Next (Próximo):

   - Stack name (Nome da pilha): escolha um nome de pilha para a pilha do AWS CloudFormation. Por exemplo, você pode chamar de ***al-nodes***. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo dentro da Região da AWS e da Conta da AWS na qual você está criando o cluster.
   - ClusterName: insira o nome do cluster. Se esse nome não corresponder ao nome do cluster, os nós não poderão se integrar ao cluster.
   - ClusterControlPlaneSecurityGroup: escolha o valor de SecurityGroups no resultado do AWS CloudFormation que você gerou na criação da VPC.

     As etapas a seguir mostram uma operação para recuperar o grupo aplicável.

     1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
     2. Escolha o nome do cluster.
     3. Escolha a guia Redes.

4. Use o valor de Additional security group (Grupos de segurança adicionais) como referência ao selecionar na lista suspensa ClusterControlPlaneSecurityGroup.

- NodeGroupName: insira um nome para o grupo de nós. Esse nome poderá ser usado superiormente para identificar o grupo de nós do Auto Scaling que for criado para os nós.
- NodeAutoScalingGroupMinSize: digite o número mínimo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeAutoScalingGroupDesiredCapacity: insira o número desejado de nós para o qual dimensionar quando a pilha for criada.
- NodeAutoScalingGroupMaxSize: digite o número máximo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeInstanceType: escolha um tipo de instância para os nós. Se o cluster estiver sendo executado na Nuvem AWS, para obter mais informações, consulte Escolher um tipo de instância de nó do Amazon EC2 ideal. Se o cluster estiver sendo executado em um Outpost, você só poderá selecionar um tipo de instância que esteja disponível no Outpost.
- NodeImageIdSSMParam: pré-preenchido com o parâmetro do Amazon EC2 Systems Manager de uma AMI recente otimizada para Amazon EKS, para uma versão do Kubernetes variável. Para usar uma versão secundária diferente do Kubernetes compatível com o Amazon EKS, substitua *1.XX* por uma versão diferente compatível. Recomendamos especificar a mesma versão do Kubernetes que seu cluster.

  Para usar a AMI acelerada otimizada para o Amazon EKS, substitua o *amazon-linux-2* por **amazon-linux-2-gpu**. Para usar a AMI Arm otimizada para o Amazon EKS, substitua *amazon-linux-2* por **amazon-linux-2-arm64**.

  > **Note**
  >
  > A AMI do nó do Amazon EKS é baseada no Amazon Linux. Você pode monitorar eventos de segurança ou privacidade para o Amazon Linux 2 no Centro de segurança do Amazon Linux ou fazendo a assinatura do feed RSS associado. Os eventos de segurança e privacidade incluem uma visão geral do problema, quais pacotes são afetadas e como atualizar suas instâncias para corrigir o problema.

- NodeImageId: (opcional) se estiver usando sua própria AMI personalizada (em vez da AMI otimizada para o Amazon EKS), insira um ID de AMI do nó para a sua Região da AWS. Se você especificar um valor aqui, ele substituirá todos os valores no campo NodeImageIdSSMParam.

- NodeVolumeSize: especifique um tamanho de volume raiz para os nós, em GiB.
- NodeVolumeType: especifique um tipo de volume raiz para os nós.
- KeyName: insira o nome de um par de chaves SSH do Amazon EC2; que você pode usar para conectar-se usando SSH em seus nós, depois de iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no AWS Management Console. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2.

  > **Note**
  >
  > Se você não fornecer um par de chaves aqui, ocorrerá uma falha ao criar a pilha do AWS CloudFormation.

- BootstrapArguments: há vários argumentos opcionais que você pode transmitir para os nós. Para obter mais informações, consulte as bootstrap script usage information (informações sobre o uso do script de boostrap) no GitHub. Se você estiver adicionando nós a um cluster local do Amazon EKS no AWS Outposts (em que as instâncias do ambiente de gerenciamento do Kubernetes são executadas no AWS Outposts) e o cluster não tiver conexão de Internet de entrada e saída (também conhecidos como clusters privados), você deverá fornecer os argumentos de bootstrap a seguir (como uma única linha).

  ```
  --b64-cluster-ca ${CLUSTER_CA} --apiserver-endpoint https://
  ${APISERVER_ENDPOINT} --enable-local-outpost true --cluster-id ${CLUSTER_ID}
  ```

- DisableIMDSv1: por padrão, cada nó oferece suporte ao serviço de metadados de instância versão 1 (IMDSv1) e IMDSv2. Você pode desabilitar IMDSv1. Para evitar que futuros nós e Pods do grupo de nós usem IMDSv1, defina DisableIMDSv1 como true (verdadeiro). Para obter mais informações, consulte Configuring the instance metadata service (Configurar o serviço de metadados de instância). Para obter mais informações sobre como restringir o acesso a ele em seus nós, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.
- VpcId: insira o ID da VPC que você criou. Antes de escolher uma VPC, revise Requisitos e considerações para VPCs.
- Sub-redes:se o cluster estiver em um Outpost, escolha pelo menos uma sub-rede privada na VPC. Antes de escolher as sub-redes, analise os Requisitos e considerações para sub-

redes. É possível ver quais sub-redes são privadas abrindo cada link de sub-rede na guia Networking (Redes) do cluster.

6. Selecione as opções desejadas na página Configure stack options (Configurar opções de pilha) e depois escolha Next (Próximo).
7. Marque a caixa de seleção à esquerda de I acknowledge that AWS CloudFormation might create IAM resources. (Reconheço que o pode criar recursos do IAM) e escolha Create stack (Criar pilha).
8. Quando a criação da pilha for concluída, selecione-a no console e escolha Outputs (Saídas).
9. Registre o valor de NodeInstanceRole para o grupo de nós criado. Você precisará dele ao configurar os nós do Amazon EKS.

Etapa 2: Habilitar os nós a ingressar no cluster

1. Verifique se você já tem um `aws-auth` `ConfigMap`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se você receber um `aws-auth` `ConfigMap`, atualize-o conforme necessário.

   a. Abra o `ConfigMap` para edição.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Adicione uma nova `mapRoles` entrada conforme necessário. Defina o valor `rolearn` para o valor NodeInstanceRole que você registrou no procedimento anterior.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

   c. Salve o arquivo e saia do seu editor de texto.

3. Se você recebeu um erro informando "`Error from server (NotFound): configmaps "aws-auth" not found`, aplique o estoque`ConfigMap`.

   a. Faça download do mapa de configuração.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-
   eks/cloudformation/2020-10-29/aws-auth-cm.yaml
   ```

   b. No arquivo `aws-auth-cm.yaml`, configure `rolearn` para o valor NodeInstanceRole que você registrou no procedimento anterior. É possível fazer isso com um editor de texto ou substituindo *`my-node-instance-role`* e executando o seguinte comando:

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-
   instance-role|' aws-auth-cm.yaml
   ```

   c. Aplique a configuração. Esse comando pode demorar alguns minutos para ser concluído.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

4. Observe o status de seus nós e aguarde até que eles atinjam o status `Ready`.

   ```
   kubectl get nodes --watch
   ```

   Insira `Ctrl+C` para retornar a um prompt de shell.

   > **Note**
   >
   > Se você receber qualquer erro de autorização ou de tipo de recurso, consulte Acesso negado ou não autorizado (`kubectl`) no tópico de solução de problemas.

   Se os nós não se integrarem ao cluster, consulte Worker nodes fail to join cluster (Falha nos nós ao ingressar no cluster) em Solucionar problemas com clusters e nós do Amazon EKS e Não é possível integrar nós a um cluster em Solucionar problemas de clusters locais do Amazon EKS no AWS Outposts.

5. Instale o driver da CSI do Amazon EBS. Para obter mais informações, consulte Instalação no GitHub. Na seção Set up driver permission (Configurar permissão do driver), certifique-

se de seguir as instruções para a opção Using IAM instance profile (Usar o perfil de instância do IAM). Você deve usar a classe de armazenamento gp2. A classe de armazenamento gp3 não é compatível.

Para criar uma classe de armazenamento gp2 no cluster, conclua as etapas a seguir.

1. Execute o seguinte comando para criar o arquivo `gp2-storage-class.yaml`.

```
cat >gp2-storage-class.yaml <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp2
  encrypted: "true"
allowVolumeExpansion: true
EOF
```

2. Aplique o manifesto ao cluster.

```
kubectl apply -f gp2-storage-class.yaml
```

6. (Somente nós de GPU) Se tiver escolhido um tipo de instância de GPU e a AMI acelerada otimizada para o Amazon EKS, você deverá aplicar o Plug-in de dispositivo NVIDIA para Kubernetes como um DaemonSet no cluster. Substitua *vX.X.X* pela versão desejada de NVIDIA/k8s-device-plugin antes de executar o comando a seguir.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-
plugin/vX.X.X/deployments/static/nvidia-device-plugin.yml
```

Etapa 3: Ações adicionais

1. (Opcional) Implante uma aplicação de amostra para testar o cluster e os nós do Linux.
2. Se o cluster estiver implantado em um Outpost, pule esta etapa. Se o cluster estiver implantado na Nuvem AWS, as informações a seguir são opcionais. Se a política gerenciada

do IAM AmazonEKS_CNI_Policy estiver anexada ao Perfil do IAM em nós do Amazon EKS, recomendamos atribuí-la a um perfil do IAM que você associa à conta de serviço Kubernetes `aws-node` do Kubernetes. Para obter mais informações, consulte Configuração do Amazon VPC CNI plugin for Kubernetes a fim de usar perfis do IAM para contas de serviço (IRSA).

Estender os recursos do Amazon EKS com projetos de código aberto

Esses projetos de código aberto ampliam a funcionalidade dos clusters do Kubernetesem execução dentro ou fora da AWS, incluindo clusters gerenciados pelo Amazon EKS.

Ferramentas de gerenciamento

Ferramentas de gerenciamento relacionadas para clusters do Amazon EKS e do Kubernetes.

eksctl

O `eksctl` é uma simples ferramenta de CLI para criar clusters no Amazon EKS.

- URL do projeto
- Documentação do projeto
- Blog de código aberto da AWS: eksctl: cluster do Amazon EKS com um comando

Controladores da AWS para Kubernetes

Com controladores da AWS para Kubernetes, você pode criar e gerenciar os recursos da AWS diretamente no cluster do Kubernetes.

- URL do projeto
- Blog sobre código aberto da AWS: operador de serviço da AWS para Kubernetes agora disponível

Flux CD

Flux é uma ferramenta que você pode usar para gerenciar a configuração do cluster usando o Git. Ele usa um operador no cluster para disparar implantações dentro do Kubernetes. Para obter mais informações sobre operadores, consulte OperatorHub.io no GitHub.

- URL do projeto
- Documentação do projeto

CDK para Kubernetes

Com o CDK para Kubernetes (cdk8s), você pode definir as aplicações e os componentes do Kubernetes usando linguagens de programação familiares. As aplicações cdk8s sintetizam em manifestos padrão do Kubernetes, que podem ser aplicados a qualquer cluster do Kubernetes.

- URL do projeto
- Documentação do projeto
- Blog sobre contêineres da AWS: Introducing cdk8s+: Intent-driven APIs for Kubernetes objects (Apresentação das cdk8s+: APIs direcionadas por intenção para objetos do Kubernets)

Redes

Projetos de redes relacionados para clusters do Amazon EKS e do Kubernetes.

Amazon VPC CNI plugin for Kubernetes

O Amazon EKS é compatível com redes VPC nativas por meio do Amazon VPC CNI plugin for Kubernetes. O plug-in atribui um endereço IP da VPC para cada Pod.

- URL do projeto
- Documentação do projeto

AWS Load Balancer Controller para Kubernetes

O AWS Load Balancer Controller ajuda a gerenciar AWS Elastic Load Balancers para um cluster do Kubernetes. Ele satisfaz os recursos do Kubernetes Ingress provisionando AWS Application Load Balancers. Ele satisfaz os recursos do serviço Kubernetes provisionando AWS Network Load Balancers.

- URL do projeto
- Documentação do projeto

ExternalDNS

O ExternalDNS sincroniza os serviços e ingressos do Kubernetes expostos com provedores DNS incluindo o Amazon Route 53 e o AWS Service Discovery.

- URL do projeto
- Documentação do projeto

Machine learning

Projetos de machine learning relacionados para clusters do Amazon EKS e do Kubernetes.

Kubeflow

Um toolkit de machine learning para o Kubernetes.

- URL do projeto
- Documentação do projeto
- Blog de código aberto da AWS: Kubeflow no Amazon EKS

Auto Scaling

Projetos de escalação automática relacionados para clusters do Amazon EKS e do Kubernetes.

Cluster autoscaler

O Cluster Autoscaler é uma ferramenta que ajusta automaticamente o tamanho do cluster do Kubernetes com base na CPU e na pressão de memória.

- URL do projeto
- Documentação do projeto
- Workshop do Amazon EKS: escalador automático de clusters

Karpenter

O Karpenter é um escalador automático de nós do Kubernetes criado para oferecer flexibilidade, performance e simplicidade.

- URL do projeto
- Documentação do projeto
- Workshop do Amazon EKS: Karpenter

Escalator

O Escalator é um autoscaler horizontal otimizado para tarefa ou lote para o Kubernetes.

- URL do projeto
- Documentação do projeto

Monitorar

Projetos de monitoramento relacionados para clusters do Amazon EKS e do Kubernetes.

Prometheus

O Prometheus é um toolkit de código aberto para monitoramento de sistemas e emissão de alertas.

- URL do projeto
- Documentação do projeto
- Workshop o Amazon EKS: https://eksworkshop.com/intermediate/240_monitoring/

Integração contínua/implantação contínua

Projetos de CI/CD relacionados para clusters do Amazon EKS e do Kubernetes.

Jenkins X

Solução de CI/CD para aplicações de nuvem modernas nos clusters do Amazon EKS e do Kubernetes.

- URL do projeto
- Documentação do projeto

Saiba mais sobre os novos recursos e o roteiro do Amazon EKS

Você pode saber mais sobre os novos recursos do Amazon EKS rolando para o feed Novidades na página de Novidades da AWS. Você também pode revisar o roadmap (roteiro) no GitHub, que informa sobre os recursos e prioridades a serem lançados em breve para que você possa planejar como deseja usar o Amazon EKS no futuro. Você pode nos fornecer feedback direto sobre as prioridades do roteiro.

Histórico do documento

A tabela a seguir descreve as principais atualizações e novos recursos para o manual do usuário do Amazon EKS. Também atualizamos a documentação com frequência para abordar os comentários enviados por você.

Alteração	Descrição	Data
Atualizações de conteúdo baseadas em cenários	Renomeamos e atualizamos os tópicos para torná-los mais orientados por cenários ao longo de todo o guia.	9 de agosto de 2024
Endpoints de interface da VPC de pilha dupla para o Amazon EKS	Agora você pode criar endpoints de interface da VPC de pilha dupla para o Amazon EKS com endereços IP e nomes DNS `IPv4` e `IPv6`.	7 de agosto de 2024
Novos endpoints de pilha dupla para as APIs do Amazon EKS com endereços `IPv6`	A API do EKS para criar e gerenciar clusters e os URLs do emissor OIDC para clusters têm novos endpoints de pilha dupla. O novo nome DNS da API do Amazon EKS é `eks.`*`region`*`.api.aws`, o qual é resolvido em endereços `IPv4` e endereços `IPv6`. Os novos clusters têm um novo URL de emissor OIDC de pilha dupla (`oidc-eks.`*`region`*`.api.aws`).	1º de agosto de 2024
Blocos de capacidade para grupos de nós gerenciados	Você agora pode usar blocos de capacidade para grupos de nós gerenciados.	1.º de julho de 2024

Coleção de métricas de grupos do Auto Scaling habilitada por padrão	Os grupos de nós gerenciados pelo Amazon EKS agora têm as métricas de grupo do Amazon EC2 Auto Scaling habilitadas por padrão, sem custo adicional. Anteriormente, era necessário executar várias etapas para habilitar esse recurso.	28 de junho de 2024
Atualizações de políticas gerenciadas da AWS: atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	27 de junho de 2024
Kubernetes versão `1.26`	A versão `1.26` do Kubernetes agora está em suporte estendido.	12 de junho de 2024
Melhorias nas referências de informações da AMI	Fizemos melhorias nas referências de informações da AMI, especialmente para Bottlerocket.	12 de junho de 2024
Kubernetes versão `1.30`	Suporte adicionado ao Kubernetes versão `1.30` para novos clusters e atualizações de versão.	23 de maio de 2024
Atualização da versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança. Isso inclui novas versões de patch do Kubernetes `1.29.4`, `1.28.9` e `1.27.13`.	14 de maio de 2024

Autoescalabilidade do CoreDNS	O mecanismo de autoescalabilidade do CoreDNS adaptará dinamicamente o número de réplicas da implantação do CoreDNS em um cluster do EKS com base no número de nós e núcleos de CPU. Esse recurso funciona para o CoreDNS `v1.9` e a versão da plataforma mais recente do release do EKS versão `1.25` e posteriores.	14 de maio de 2024
Suporte ao CloudWatch Container Insights para Windows	O complemento Amazon CloudWatch Observability Operator agora também permite Container Insights em nós de processamento Windows no cluster.	10 de abril de 2024
Conceitos do Kubernetes	Adição de um novo tópico sobre conceitos do Kubernetes.	5 de abril de 2024
Reestruturação do conteúdo sobre acesso e IAM	Mova as páginas existentes relacionadas aos tópicos de acesso e IAM, como mapa de configuração de autenticação, entradas de acesso, ID do pod e IRSA, para a nova seção. Revise o conteúdo da visão geral.	2 de abril de 2024
Suporte ao sistema operacional Bottlerocket para o driver da CSI do Amazon S3	Agora o driver da CSI Mountpoint para Amazon S3 é compatível com o Bottlerocket.	13 de março de 2024

Atualizações de política gerenciada pela AWS: atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	4 de março de 2024
Amazon Linux 2023	O Amazon Linux 2023 (AL2023) é um novo sistema operacional baseado no Linux que foi projetado para fornecer um ambiente seguro, estável e de alta performance para as aplicações em nuvem.	29 de fevereiro de 2024
Identidade de Pods do EKS e do IRSA compatível com arquivos associados no Kubernetes1.29	No Kubernetes 1.29, os contêineres auxiliares estão disponíveis nos clusters do Amazon EKS. Há suporte para os contêineres associados com o uso de perfis do IAM para contas de serviço ou da Identidade de Pods do EKS. Para obter mais informaçõ es sobre auxiliares, consulte Sidecar Containers na documentação do Kubernetes.	26 de fevereiro de 2024
Kubernetes versão 1.29	Suporte adicionado ao Kubernetes versão 1.29 para novos clusters e atualizações de versão.	23 de janeiro de 2024
Versão completa: Amazon EKS Extended Support para versões do Kubernetes	O suporte de Kubernetes versão estendida permite que você permaneça em uma Kubernetes versão específica por mais de 14 meses.	16 de janeiro de 2024

Detecção da integridade do cluster do Amazon EKS na Nuvem AWS	O Amazon EKS detecta problemas com os clusters do Amazon EKS e com a infraestrutura dos pré-requisitos do cluster na integridade do cluster. É possível visualizar os problemas com os clusters do EKS no AWS Management Console e na `health` do cluster na API do EKS. Esses problemas são adicionais aos problemas detectados e exibidos pelo console. Anteriormente, a integridade do cluster só estava disponível para clusters locais no AWS Outposts.	28 de dezembro de 2023
Expansão Amazon EKS da Região da AWS	O Amazon EKS já está disponível na Região da AWS Oeste do Canadá (Calgary) (`ca-west-1`).	20 de dezembro de 2023
Insights sobre clusters	Agora você pode obter recomendações sobre seu cluster com base em verificações recorrentes.	20 de dezembro de 2023

Agora você pode conceder aos perfis e usuários do IAM acesso ao cluster usando entradas de acesso	Antes da introdução das entradas de acesso, o acesso ao cluster era concedido aos perfis e usuários do IAM ao adicionar entradas ao `ConfigMap aws-auth`. Agora, cada cluster tem um modo de acesso e você pode passar a usar entradas de acesso em sua agenda. Depois de alternar os modos, você pode adicionar usuários ao adicionar entradas de acesso na AWS CLI, AWS CloudFormation e SDKs da AWS.	18 de dezembro de 2023
Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança . Isto inclui novas versões de patch do Kubernetes `1.28.4`, `1.27.8`, `1.26.11` e `1.25.16`.	12 de dezembro de 2023
Mountpoint para driver da CSI do Amazon S3	Agora é possível instalar o Mountpoint para driver da CSI do Amazon S3 para OpenZFS em clusters do Amazon EKS.	27 de novembro de 2023

Ativar as métricas do Prometheus ao criar um cluster	No AWS Management Console, agora é possível ativar as métricas do Prometheus ao criar um cluster. Você também pode ver os detalhes do extrator do Prometheus na guia Observabilidade.	26 de novembro de 2023
Amazon EKS Pod Identities	Os Amazon EKS Pod Identities associam um perfil do IAM a uma conta de serviço da Kubernetes. Com esse recurso, você não precisa mais fornecer permissões estendidas ao perfil do IAM do nó. Assim sendo, os Pods nesse nó podem chamar APIs da AWS. Diferentemente dos perfis do IAM para contas de serviço, os EKS Pod Identities estão completamente dentro do EKS; você não precisa de um provedor de identidade OIDC.	26 de novembro de 2023
Atualizações de política gerenciada pela AWS: atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	26 de novembro de 2023
Controlador de snapshots da CSI	Agora é possível instalar o controlador de snapshots da CSI para uso com drivers da CSI compatíveis, como o driver da CSI do Amazon EBS.	17 de novembro de 2023

Reescrever o tópico do ADOT Operator	O suporte a complementos do Amazon EKS para a seção ADOT Operator era redundant e com a documentação do AWS Distro para OpenTelem etry. Migramos as informações essenciais restantes para esse recurso para reduzir informaçõ es desatualizadas e inconsist entes.	14 de novembro de 2023
Suporte ao complemen to CoreDNS do EKS para métricas do Prometheus	As versões `v1.10.1-e ksbuild.5` , `v1.9.3-ek sbuild.9` e `v1.8.7-ek sbuild.8` do complemen to do EKS para CoreDNS expõem a porta na qual o CoreDNS publicou as métricas no serviço `kube-dns`. Isso facilita a inclusão das métricas do CoreDNS em seus sistemas de monitoram ento.	10 de novembro de 2023
Complemento Amazon EKS CloudWatch Observability Operator	Adição da página Amazon EKS CloudWatch Observabi lity Operator	6 de novembro de 2023
Blocos de capacidade para instâncias P5 autogerenciadas no Leste dos EUA (Ohio)	No Leste dos EUA (Ohio), agora é possível usar blocos de capacidade para instâncias P5 autogerenciadas.	31 de outubro de 2023

Clusters oferecem suporte à modificação de sub-redes e grupos de segurança	Você pode atualizar o cluster para alterar quais sub-redes e grupos de segurança o cluster usa. Você pode atualizar AWS Management Console a partir da versão mais recente da versão AWS CLI, AWS CloudFormation e `eksctl` `v0.164.0-rc.0` ou posterior. Talvez seja necessário fazer isso para fornecer às sub-redes mais endereços IP disponíveis para atualizar com sucesso uma versão de cluster.	24 de outubro de 2023
A função de cluster e a função de grupo de nós gerenciados oferecem suporte AWS Identity and Access Management às políticas gerenciadas pelo cliente	Você pode usar uma política personalizada do IAM na função do cluster, em vez da `AmazonEKSClusterPolicy` AWS política gerenciada. Além disso, você pode usar uma política personalizada do IAM na função do nó em um grupo de nós gerenciados, em vez da `AmazonEKSWorkerNodePolicy` AWS política gerenciada. Faça isso para criar uma política com privilégio mínimo para atender aos rígidos requisitos de conformidade.	23 de outubro de 2023
Corrija o link para a instalação do eksctl	Corrija o link de instalação do eksctl depois que a página foi movida.	6 de outubro de 2023

Versão prévia: Amazon EKS Extended Support para Kubernetes versões	O suporte de Kubernetes versão estendida permite que você permaneça em uma Kubernetes versão específica por mais de 14 meses.	4 de outubro de 2023
Remover referências à AWS App Mesh integração	As integrações do Amazon EKS AWS App Mesh permanecem somente para clientes atuais do App Mesh.	29 de setembro de 2023
Kubernetes versão `1.28`	Suporte adicionado ao Kubernetes versão `1.28` para novos clusters e atualizações de versão.	26 de setembro de 2023
Os clusters existentes oferecem suporte Kubernetes à aplicação de políticas de rede no Amazon VPC CNI plugin for Kubernetes	Você pode usar a Kubernetes política de rede em clusters existentes com o Amazon VPC CNI plugin for Kubernetes, em vez de precisar de uma solução de terceiros.	15 de setembro de 2023
O complemento CoreDNS Amazon EKS oferece suporte à modificação do PDB	Você pode modificar o `PodDisruptionBudget` complemento EKS CoreDNS em versões `v1.9.3-eksbuild.7` e posteriores `v1.10.1-eksbuild.4` e posteriores.	15 de setembro de 2023
Compatibilidade do Amazon EKS com sub-redes compartilhadas	Novos requisitos e considerações sobre sub-redes compartilhadas para criar clusters do Amazon EKS em sub-redes compartilhadas.	7 de setembro de 2023

Atualizações de O que é o Amazon EKS?	Novos tópicos foram adicionados, Casos de uso comuns e Arquitetura. Outros tópicos foram atualizados.	6 de setembro de 2023
Aplicação da política de rede do Kubernetes no Amazon VPC CNI plugin for Kubernetes	Você pode usar a política de rede do Kubernetes com o Amazon VPC CNI plugin for Kubernetes em vez de precisar de uma solução de terceiros.	29 de agosto de 2023
Expansão da Região da AWS do Amazon EKS	Agora, o Amazon EKS está disponível na Região da AWS de Israel (Tel Aviv) (`il-central-1`).	1.º de agosto de 2023
Armazenamento temporário configurável para o Fargate	Você pode aumentar a quantidade total de armazenamento temporário para cada Pod em execução no Amazon EKS Fargate.	31 de julho de 2023
Suporte ao complemento para o driver da CSI do Amazon EFS	Agora, você pode usar o AWS Management Console, a AWS CLI e a API para gerenciar o driver da CSI do Amazon EFS.	26 de julho de 2023
Atualizações de políticas gerenciadas da AWS - Nova política	O Amazon EKS adicionou uma nova política gerenciada da AWS.	26 de julho de 2023
As atualizações de versão do Kubernetes para 1.27, 1.26, 1.25 e 1.24 já estão disponíveis para clusters locais no AWS Outposts	As atualizações de versão do Kubernetes para 1.27.3, 1.26.6, 1.25.11 e 1.24.15 já estão disponíveis para clusters locais no AWS Outposts	20 de julho de 2023

Suporte a prefixos IP para nós do Windows	A atribuição de prefixos IP aos seus nós pode permitir a hospedagem de um número significativamente maior de Pods em seus nós do que a atribuição de endereços IP secundários individuais aos seus nós.	6 de julho de 2023
Driver da CSI do Amazon FSx para OpenZFS	Agora é possível instalar o driver da CSI do Amazon FSx para OpenZFS em clusters do Amazon EKS.	30 de junho de 2023
Os Pods em nós do Linux em clusters `IPv4` agora podem se comunicar com endpoints `IPv6`.	Após a atribuição de um endereço IPv6 ao seu nó, o endereço de rede `IPv4` dos seus Pods é traduzido para o endereço `IPv6` do nó em que ele está sendo executado.	19 de junho de 2023
Grupos de nós gerenciados do Windows em AWS GovCloud (US) Regions	Agora, nas AWS GovCloud (US) Regions, os grupos de nós gerenciados do Amazon EKS podem executar contêineres do Windows.	30 de maio de 2023
Kubernetes versão `1.27`	Suporte adicionado ao Kubernetes versão `1.27` para novos clusters e atualizações de versão.	24 de maio de 2023
Kubernetes versão `1.26`	Suporte adicionado ao Kubernetes versão `1.26` para novos clusters e atualizações de versão.	11 de abril de 2023

gMSA sem domínio	Agora, você pode usar o gMSA sem domínio com Windows Pods.	27 de março de 2023
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível na Região da AWS Ásia-Pacífico (Melbourne) (`ap-southeast-4`).	10 de março de 2023
Driver da CSI do Amazon File Cache	Atualmente, é possível instalar o driver da CSI do Amazon File Cache em clusters do Amazon EKS.	3 de março de 2023
KubernetesA versão 1.25 do já está disponível para clusters locais no AWS Outposts.	Você já pode criar um cluster local do Amazon EKS em um Outpost usando o Kubernetes versões `1.22` a `1.25`.	1.º de março de 2023
Kubernetes versão `1.25`	Suporte adicionado ao Kubernetes versão `1.25` para novos clusters e atualizações de versão.	22 de fevereiro de 2023
Atualizações de política gerenciada pela AWS: atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	7 de fevereiro de 2023
Expansão da Região da AWS do Amazon EKS	O Amazon EKS agora está disponível na Ásia-Pacífico (Hyderabad) (`ap-south-2`), Europa (Zurique) (`eu-central-2`) e Europa (Espanha) (`eu-south-2`) Regiões da AWS.	6 de fevereiro de 2023

Agora, as versões `1.21` a `1.24` do Kubernetes estão disponíveis para clusters locais no AWS Outposts.	Você já pode criar um cluster local do Amazon EKS em um Outpost usando o Kubernetes versões `1.21` a `1.24`. Anteriormente, apenas a versão `1.21` estava disponível.	17 de janeiro de 2023
O Amazon EKS agora é compatível com o AWS PrivateLink	Você pode usar um AWS PrivateLink para criar uma conexão privada entre a sua VPC e o Amazon EKS.	16 de dezembro de 2022
Compatibilidade do Windows com grupos de nós gerenciados	Você agora pode usar o Windows para os grupos de nós gerenciados do Amazon EKS.	15 de dezembro de 2022
Os complementos do Amazon EKS de fornecedores de software independentes agora estão disponíveis no AWS Marketplace	Agora você pode navegar e assinar os complementos do Amazon EKS de fornecedores de software independentes por meio do AWS Marketplace.	28 de novembro de 2022
Atualizações de política gerenciada pela AWS: atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	17 de novembro de 2022
Kubernetes versão `1.24`	Suporte adicionado ao Kubernetes versão `1.24` para novos clusters e atualizações de versão.	15 de novembro de 2022
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível na Região da AWS Oriente Médio (UAE) (`me-central-1`).	3 de novembro de 2022

Atualizações de política gerenciada pela AWS: atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	24 de outubro de 2022
Atualizações de política gerenciada pela AWS: atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	20 de outubro de 2022
Clusters locais no AWS Outposts agora estão disponíveis	Agora, você pode criar um cluster local do Amazon EKS em um Outpost.	19 de setembro de 2022
Cotas baseadas em vCPU do Fargate	O Fargate está fazendo a transição de cotas baseadas em Pod para cotas baseadas em vCPU.	8 de setembro de 2022
Atualizações de política gerenciada pela AWS: atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	31 de agosto de 2022
Monitoramento de custos	O Amazon EKS agora oferece suporte para o Kubecost, que permite monitorar os custos detalhados por recursos do Kubernetes, incluindo Pods, nós, namespaces e rótulos.	24 de agosto de 2022
Atualizações de políticas gerenciadas da AWS - Nova política	O Amazon EKS adicionou uma nova política gerenciada da AWS.	24 de agosto de 2022
Atualizações de políticas gerenciadas da AWS - Nova política	O Amazon EKS adicionou uma nova política gerenciada da AWS.	23 de agosto de 2022

Recursos de tag para faturamento	Adicionada tag de alocação de custos gerada por `aws:eks:cluster-name` para todos os clusters.	16 de agosto de 2022
Curingas do perfil do Fargate	Foi adicionado suporte para curingas do perfil do Fargate nos critérios do seletor para namespaces, chaves de rótulo e valores de rótulos.	16 de agosto de 2022
Kubernetes versão `1.23`	Suporte adicionado ao Kubernetes versão `1.23` para novos clusters e atualizações de versão.	11 de agosto de 2022
Visualizar recursos do Kubernetes no AWS Management Console	Agora, é possível ver informações sobre os recursos do Kubernetes implantados no seu cluster utilizando o AWS Management Console.	3 de maio de 2022
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível na Região da AWS Ásia-Pacífico (Jacarta) (`ap-southeast-3`).	2 de maio de 2022
Página de observabilidade e suporte ao complemento ADOT	Adicionada página sobre capacidade de observação e o AWS Distro for OpenTelemetry (ADOT).	21 de abril de 2022
Kubernetes versão `1.22`	Suporte adicionado ao Kubernetes versão `1.22` para novos clusters e atualizações de versão.	4 de abril de 2022

Atualizações de políticas gerenciadas da AWS - Nova política	O Amazon EKS adicionou uma nova política gerenciada da AWS.	4 de abril de 2022
Adicionados detalhes sobre a aplicação de patches em Pod do Fargate	Ao fazer upgrade de Pods do Fargate, o Amazon EKS primeiro tenta remover os Pods com base nos orçamentos de interrupção dos Pod. É possível criar regras de eventos para reagir a remoções com falha antes que os Pods sejam excluídos.	1.º de abril de 2022
Lançamento completo: suporte ao complemento para o driver da CSI do Amazon EBS	Agora você pode usar o AWS Management Console, a AWS CLI e a API para gerenciar o driver da CSI do Amazon EBS.	31 de março de 2022
Atualização de conteúdo do AWS Outposts	Instruções para implantar um cluster do Amazon EKS no AWS Outposts.	22 de março de 2022
Atualizações de política gerenciada pela AWS: atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	21 de março de 2022
Suporte ao Windows `containerd`	Agora, é possível selecionar o runtime do `containerd` para nós do Windows.	14 de março de 2022
Adição de considerações sobre o Conector do Amazon EKS à documentação de segurança	Descreve o modelo de responsabilidade compartil hada no que diz respeito a clusters conectados.	25 de fevereiro de 2022

Atribuir endereços `IPv6` aos Pods e serviços	Agora você pode criar um cluster `1.21` ou posterior que atribua endereços `IPv6` aos Pods e serviços.	6 de janeiro de 2022
Atualizações de política gerenciada pela AWS: atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	13 de dezembro de 2021
Lançamento da versão de demonstração: suporte ao complemento para o Amazon EBS CSI	Agora, você pode visualizar usando o AWS Management Console, a AWS CLI e a API para gerenciar o driver da CSI do Amazon EBS.	9 de dezembro de 2021
Suporte ao escalador automático Karpenter	Agora você pode usar o projeto de código aberto Karpenter para escalar seus nós automaticamente.	29 de novembro de 2021
Suporte ao filtro Fluent Bit Kubernetes no registro do Fargate	Agora você pode usar o filtro Fluent Bit Kubernetes com o registro do Fargate.	10 de novembro de 2021
Suporte ao Windows disponível no ambiente de gerenciamento	O suporte ao Windows já está disponível no ambiente de gerenciamento. Não é mais necessário habilitá-lo no plano de dados.	9 de novembro de 2021

Bottlerocket adicionado como um tipo de AMI para grupos de nós gerenciados	Antes, o Bottlerocket só estava disponível como uma opção de nó autogerenciado. Agora, ele pode ser configurado como um grupo de nós gerenciados, reduzindo os esforços necessários para atender aos requisitos de conformidade dos nós.	28 de outubro de 2021
Suporte ao driver DL1	As AMIs personalizadas do Amazon Linux agora oferecem suporte a workloads de aprendizado profundo para Amazon Linux 2. Essa ativação permite uma configuração de linha de base genérica on-premises ou na nuvem.	25 de outubro de 2021
Suporte a vídeo VT1	As AMIs personalizadas do Amazon Linux agora suportam VT1 para algumas distribuições. Essa habilitação anuncia dispositivos Xilinx U30 em seu cluster do Amazon EKS.	13 de setembro de 2021
O Amazon EKS Connector já está disponível	Você pode usar o Amazon EKS Connector para registrar e conectar qualquer cluster do Kubernetes à AWS compatível e visualizá-lo no console do Amazon EKS.	8 de setembro de 2021

O Amazon EKS Anywhere já está disponível	O Amazon EKS Anywhere é uma nova opção de implantação para o Amazon EKS que você pode usar para criar e operar clusters do Kubernetes on-premises.	8 de setembro de 2021
Driver de CSI do Amazon FSx for NetApp ONTAP	Adicionado tópico que resume o driver de CSI do Amazon FSx for NetApp ONTAP e fornece links para outras referências.	2 de setembro de 2021
Grupos de nós gerenciados agora calculam automaticamente o máximo recomendado de Pods do Amazon EKS para os nós	Grupos de nós gerenciados agora calculam automaticamente o máximo recomendado de Pods do Amazon EKS para os nós que você implanta sem um modelo de inicialização ou com um modelo de inicialização em que você não especificou um ID de AMI.	30 de agosto de 2021
Remover o gerenciamento de configurações de complemento do Amazon EKS sem remover o software do complemento do Amazon EKS	Agora você pode remover um complemento do Amazon EKS sem remover o software do complemento do cluster.	20 de agosto de 2021
Criar Pods com vários hospedeiros usando o Multus	Agora, você pode adicionar várias interfaces de rede a um Pod usando o Multus.	2 de agosto de 2021

Adicionar mais endereços IP aos nós do Amazon EC2 para Linux	Agora você pode adicionar significativamente mais endereços IP aos nós do Amazon EC2 para Linux. Isso significa que você pode executar uma densidade maior de Pods em cada nó.	27 de julho de 2021
Bootstrap de runtime do `containerd`	A imagem de máquina da Amazon (AMI) do Amazon Linux acelerada e otimizada para o Amazon EKS agora contém um sinalizador de bootstrap que você pode usar para habilitar o runtime do `containerd` em AMIs otimizadas para o Amazon EKS e AMIs do Bottleroc ket. Esse sinalizador está disponível em todas as versões do Kubernetes compatíveis da AMI.	19 de julho de 2021
Kubernetes versão `1.21`	Adicionado suporte ao Kubernetes versão `1.21`.	19 de julho de 2021
Tópico de políticas gerenciadas adicionado	Uma lista de todas as políticas gerenciadas do IAM no Amazon EKS e alterações feitas nessas políticas desde 17 de junho de 2021.	17 de junho de 2021
Usar grupos de segurança para Pods com o Fargate	Agora você pode usar grupos de segurança para Pods com o Fargate, além de usá-los com os nós do Amazon EC2.	1º de junho de 2021

Adição dos complementos CoreDNS e `kube-proxy` do Amazon EKS	O Amazon EKS agora pode ajudar você a gerenciar os complementos CoreDNS e `kube-proxy` do Amazon EKS para o cluster.	19 de maio de 2021
Kubernetes versão `1.20`	Suporte adicionado ao Kubernetes versão `1.20` para novos clusters e atualizações de versão.	18 de maio de 2021
Lançamento do AWS Load Balancer Controller`2.2.0`	Agora é possível usar o AWS Load Balancer Controller para criar balanceadores de carga elásticos usando destinos IP ou de instância.	14 de maio de 2021
Taints de nós para grupos de nós gerenciados	O Amazon EKS agora oferece suporte para a adição de taints de notas a grupos de nós gerenciados.	11 de maio de 2021
Criptografia de segredos para os clusters existentes	O Amazon EKS agora oferece suporte à adição de criptografia de segredos para os clusters existentes.	26 de fevereiro de 2021
Kubernetes versão `1.19`	Suporte adicionado ao Kubernetes versão `1.19` para novos clusters e atualizações de versão.	16 de fevereiro de 2021
O Amazon EKS agora é compatível com provedores de identidade OpenID Connect (OIDC) como método de autenticar usuários no cluster `1.16` ou posterior.	Os provedores de identidade OIDC podem ser usados com o AWS Identity and Access Management (IAM) ou como uma alternativa para ele.	12 de fevereiro de 2021

Visualizar recursos de nó e workload no AWS Management Console	Agora você pode visualizar detalhes sobre os nós gerenciados, autogerenciados e do Fargate, assim como as suas workloads implantadas do Kubernetes no AWS Management Console.	1º de dezembro de 2020
Implantar tipos de instância spot em um grupo de nós gerenciados	Agora você pode implantar vários tipos de instância spot ou sob demanda em um grupo de nós gerenciados.	1º de dezembro de 2020
O Amazon EKS agora pode gerenciar complementos específicos para o cluster	O próprio usuário pode gerenciar os complementos ou pode permitir que o Amazon EKS controle o lançamento e a versão de um complemento por meio da API do Amazon EKS.	1º de dezembro de 2020
Compartilhar um ALB em várias entradas	Agora é possível compartilhar um AWS Application Load Balancer entre vários ingressos do Kubernetes. No passado, você tinha que implantar um balanceador de carga da aplicação separado para cada ingresso.	23 de outubro de 2020

Suporte para o destino de IP do NLB	Agora, você pode implantar um Network Load Balancer com destinos de IP. Isso significa que você pode usar um NLB para balancear a carga de tráfego da rede para os Pods do Fargate e diretamente para os Pods executados nos nós do Amazon EC2.	23 de outubro de 2020
Kubernetes versão `1.18`	Suporte adicionado ao Kubernetes versão `1.18` para novos clusters e atualizações de versão.	13 de outubro de 2020
Especifique um bloco CIDR personalizado para atribuição de endereços IP de serviço do Kubernetes.	Agora você pode especificar um bloco CIDR personalizado a partir do qual o Kubernetes atribui os endereços IP do serviço.	29 de setembro de 2020
Atribuir grupos de segurança a Pods individuais	Agora você pode associar grupos de segurança diferentes a alguns dos Podsindividuais que são executados em muitos tipos de instância do Amazon EC2.	9 de setembro de 2020
Implantar o Bottlerocket nos seus nós	Agora você pode implantar nós que executam o Bottlerocket.	31 de agosto de 2020
A capacidade de executar nós Arm está disponível ao público em geral	Agora você pode executar nós Arm em grupos de nós gerenciados e autogerenciados.	17 de agosto de 2020

Modelos de execução de grupos de nós gerenciados e AMI personalizada	Agora você pode implantar um grupo de nós gerenciados usando um modelo de lançamento do Amazon EC2. Caso você queira, o modelo de execução poderá especificar uma AMI personalizada.	17 de agosto de 2020
Compatibilidade do EFS com o AWS Fargate	Agora você pode usar o Amazon EFS com o AWS Fargate.	17 de agosto de 2020
Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança. Isso inclui suporte UDP para serviços do tipo `LoadBalancer` ao usar Network Load Balancers com o Kubernetes versão `1.15` ou posterior. Para obter mais informações, consulte o problema Allow UDP for AWS Network Load Balancer (Permitir UDP para o Network Load Balancer) no GitHub.	12 de agosto de 2020
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível nas Regiões da AWS África (Cidade do Cabo) (`af-south-1`) e Europa (Milão) (`eu-south-1`).	6 de agosto de 2020

Métricas de uso do Fargate	O AWS Fargate fornece métricas de uso do CloudWatch que proporcionam visibilidade do uso de recursos sob demanda do Fargate por suas contas.	3 de agosto de 2020
Kubernetes versão `1.17`	Suporte adicionado ao Kubernetes versão `1.17` para novos clusters e atualizações de versão.	10 de julho de 2020
Criar e gerenciar recursos do App Mesh de dentro do Kubernetes com o controlador do App Mesh para Kubernetes	Você pode criar e gerenciar recursos do App Mesh no Kubernetes. O controlador também injeta automaticamente o proxy Envoy e os contêineres init nos Pods que você implanta.	18 de junho de 2020
O Amazon EKS agora oferece suporte para nós Inf1 do Amazon EC2	Você pode adicionar nós Inf1 do Amazon EC2 ao seu cluster.	4 de junho de 2020
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível nas Regiões da AWS AWS GovCloud (EUA-Leste) (`us-gov-east-1`) e AWS GovCloud (EUA-Oeste) (`us-gov-west-1`).	13 de maio de 2020
O Kubernetes`1.12` não é mais compatível com o Amazon EKS	O Kubernetes versão `1.12` não é mais compatível com o Amazon EKS. Atualize qualquer cluster `1.12` para a versão `1.13` ou superior para evitar interrupção do serviço.	12 de maio de 2020

Kubernetes versão 1.16	Suporte adicionado ao Kubernetes versão 1.16 para novos clusters e atualizações de versão.	30 de abril de 2020
Adicionada a função vinculada ao serviço AWSServiceRoleForAmazonEKS	Adicionada a função vinculada ao serviço AWSServiceRoleForAmazonEKS.	16 de abril de 2020
Kubernetes versão 1.15	Suporte adicionado ao Kubernetes versão 1.15 para novos clusters e atualizações de versão.	10 de março de 2020
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível nas Regiões da AWS Pequim (cn-north-1) e Ningxia (cn-northwest-1).	26 de fevereiro de 2020
Driver de CSI do FSx for Lustre	Adição de um tópico para a instalação do driver da CSI do FSx para Lustre em clusters 1.14 do Kubernetes do Amazon EKS.	23 de dezembro de 2019
Restringir o acesso à rede ao endpoint de acesso público de um cluster	Com esta atualização, você pode usar o Amazon EKS para restringir os intervalos de CIDR que podem se comunicar com o endpoint de acesso público do servidor de API do Kubernetes.	20 de dezembro de 2019

Resolver o endereço de endpoint de acesso privado para um cluster de fora de uma VPC	Com esta atualização, você pode usar o Amazon EKS para resolver o endpoint de acesso privado do servidor de API do Kubernetes de fora de uma VPC.	13 de dezembro de 2019
(Beta) Nós de instâncias do Amazon EC2 A1 Amazon EC2	Execute os nós de instâncias A1 do Amazon EC2 registrados no cluster do Amazon EKS.	4 de dezembro de 2019
Criar um cluster no AWS Outposts	O Amazon EKS oferece suporte para a criação de clusters no AWS Outposts.	3 de dezembro de 2019
AWS Fargate no Amazon EKS	Agora os clusters Kubernetes do Amazon EKS são compatíveis com a execução de Pods no Fargate.	3 de dezembro de 2019
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível na Região da AWS Canadá (Central) (`ca-central-1`).	21 de novembro de 2019
Grupos de nós gerenciados	Os grupos de nós gerenciados do Amazon EKS automatizam o provisionamento e o gerenciamento do ciclo de vida dos nós (instâncias do Amazon EC2) para clusters de Kubernetes do Amazon EKS.	18 de novembro de 2019
Atualização de versão da plataforma do Amazon EKS	Novas versões de plataforma para resolver `CVE-2019-11253`.	6 de novembro de 2019

O Kubernetes1.11 não é mais compatível com o Amazon EKS	O Kubernetes versão 1.11 não é mais compatível com o Amazon EKS. Atualize qualquer cluster 1.11 para a versão 1.12 ou superior para evitar interrupção do serviço.	4 de novembro de 2019
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível na Região da AWS América do Sul (São Paulo) (sa-east-1).	16 de outubro de 2019
Suporte do Windows	Os clusters do Amazon EKS que executam o Kubernetes versão 1.14 agora são compatíveis com workloads do Windows.	7 de outubro de 2019
Autoescalabilidade	Adicionado um capítulo para abordar alguns dos diferentes tipos de escalabilidade automática do Kubernetes com suporte nos clusters do Amazon EKS.	30 de setembro de 2019
Atualização do painel do Kubernetes	Tópico atualizado para instalar o painel do Kubernetes em clusters do Amazon EKS para usar a versão beta 2.0.	28 de setembro de 2019
Driver da CSI do Amazon EFS	Adição de um tópico para a instalação do driver da CSI do Amazon EFS em clusters 1.14 do Kubernetes do Amazon EKS.	19 de setembro de 2019

Parâmetro do Amazon EC2 Systems Manager para o ID da AMI otimizada para o Amazon EKS	Tópico adicionado para recuperar o ID da AMI otimizada para o Amazon EKS usando um parâmetro do Amazon EC2 Systems Manager. O parâmetro elimina a necessidade de pesquisar IDs de AMIs.	18 de setembro de 2019
Marcação de recursos do Amazon EKS	Você pode gerenciar a marcação dos clusters do Amazon EKS.	16 de setembro de 2019
Driver da CSI do Amazon EBS	Tópico adicionado para instalar o driver CSI do Amazon EFS em clusters do Amazon EKS do Kubernetes `1.14`.	9 de setembro de 2019
Nova AMI com patch otimizada para Amazon EKS para `CVE-2019-9512` e `CVE-2019-9514`	O Amazon EKS atualizou a AMI otimizada para Amazon EKS, a fim de abordar `CVE-2019-9512` e `CVE-2019-9514`.	6 de setembro de 2019
Anúncio de descontinuação do Kubernetes`1.11` no Amazon EKS	O Amazon EKS encerrou o suporte para o Kubernete s versão `1.11` em 4 de novembro de 2019.	4 de setembro de 2019
Kubernetes versão `1.14`	Suporte adicionado ao Kubernetes versão `1.14` para novos clusters e atualizações de versão.	3 de setembro de 2019

Perfis do IAM para contas de serviço	Com os perfis do IAM para contas de serviço em clusters do Amazon EKS, é possível associar um perfil do IAM a uma conta de serviço do Kubernetes. Com esse recurso, você não precisa mais fornecer permissões estendidas ao perfil do IAM do nó. Assim sendo, os Pods nesse nó podem chamar APIs da AWS.	3 de setembro de 2019
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível na Região da AWS Oriente Médio (Bahrein) (`me-south-1`).	29 de agosto de 2019
Atualização de versão da plataforma do Amazon EKS	Novas versões de plataform a para resolver `CVE-2019-9512` e `CVE-2019-9514`.	28 de agosto de 2019
Atualização de versão da plataforma do Amazon EKS	Novas versões de plataform a para resolver `CVE-2019-11247` e `CVE-2019-11249`.	5 de agosto de 2019
Expansão de regiões do Amazon EKS	O Amazon EKS já está disponível na Região da AWS Ásia-Pacífico (Hong Kong) (`ap-east-1`).	31 de julho de 2019

O Kubernetes `1.10` não é mais compatível com o Amazon EKS	O Kubernetes versão `1.10` não é mais compatível com o Amazon EKS. Atualize qualquer cluster `1.10` para a versão `1.11` ou superior para evitar interrupção do serviço.	30 de julho de 2019
Adicionado tópico adicionado sobre o ALB Ingress Controller	O ALB Ingress Controller para Kubernetes da AWS é um controlador que faz com que um ALB seja criado recursos de ingresso são criados.	11 de julho de 2019
Nova AMI otimizada para o Amazon EKS	A remoção de binários `kubectl` desnecessários das AMIs.	3 de julho de 2019
Kubernetes versão `1.13`	Suporte adicionado ao Kubernetes versão `1.13` para novos clusters e atualizações de versão.	18 de junho de 2019
Nova AMI com patch otimizada para Amazon EKS para `AWS-2019-005`	O Amazon EKS atualizou a AMI otimizada do Amazon EKS para lidar com as vulnerabilidades descritas em `AWS-2019-005` .	17 de junho de 2019
Anúncio de descontinuação do suporte do Kubernetes `1.10` no Amazon EKS	O Amazon EKS interrompeu o suporte para o Kubernetes versão `1.10` em 22 de julho de 2019.	21 de maio de 2019

Atualização de versão da plataforma do Amazon EKS	Nova versão da plataform a para clusters Kubernetes `1.11` e `1.10` para oferecer suporte a nomes de DNS personalizados no certifica do do `kubelet` e melhorar a performance de `etcd`.	21 de maio de 2019
Comando da AWS CLI`get-token`	O comando `aws eks get-token` foi adicionado à AWS CLI. Você não precisa mais instalar o AWS IAM Authentic ator para Kubernetes para criar tokens de segurança de clientes para a comunicaç ão com servidor da API do cluster. Atualize a instalaçã o da AWS CLI para a versão mais recente para usar essa nova funcionalidade. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface.	10 de maio de 2019
Conceitos básicos da `eksctl`	Este guia de conceitos básicos ajuda a instalar todos os recursos necessários para começar a usar o Amazon EKS usando `eksctl`. Trata-se de um simples utilitário de linha de comando para criar e gerenciar clusters do Kubernetes no Amazon EKS.	10 de maio de 2019

Atualização de versão da plataforma do Amazon EKS	Nova versão da plataform a para clusters Kubernetes `1.12` para oferecer suporte a nomes de DNS personali zados no certificado do `kubelet` e melhorar a performance de `etcd`. Isso corrige um erro que fazia com que os daemons do `kubelet` do nó solicitassem um novo certificado a cada poucos segundos.	8 de maio de 2019
Tutorial do Prometheus	Tópico adicionado para implantação do Prometheus no cluster do Amazon EKS.	5 de abril de 2019
Registro em log do ambiente de gerenciamento do Amazon EKS	Com essa atualização, você pode obter logs de auditoria e diagnóstico diretamente do painel de controle do Amazon EKS. Você pode usar esses CloudWatch Logs em sua conta como referência para proteger e executar os clusters.	4 de abril de 2019
Kubernetes versão `1.12`	Suporte adicionado ao Kubernetes versão `1.12` para novos clusters e atualizações de versão.	28 de março de 2019
Adicionado o Guia de conceitos básicos do App Mesh	Documentação adicionada para os conceitos básicos do App Mesh e Kubernetes.	27 de março de 2019

Acesso privado ao endpoint do servidor de API do Amazon EKS	Documentação adicionada para desabilitar o acesso público para o endpoint do servidor de API do Kubernetes do cluster do Amazon EKS.	19 de março de 2019
Adicionado tópico sobre como instalar o servidor de métricas do Kubernetes	O servidor de métricas do Kubernetes é um agregador de dados de uso de recursos no cluster.	18 de março de 2019
Adição da lista de projetos de código aberto relacionados	Esses projetos de código aberto ampliam a funcionalidade de clusters do Kubernetes em execução na AWS, incluindo clusters gerenciados pelo Amazon EKS.	15 de março de 2019
Adição de tópico sobre como instalar Helm localmente	O gerenciador de pacotes `helm` para Kubernetes ajuda a instalar e gerenciar aplicações no cluster Kubernetes. Esse tópico mostra como instalar e executar os binários do `helm` e `tiller` localmente. Dessa maneira, é possível instalar e gerenciar gráficos utilizando a CLI do Helm no seu sistema local.	11 de março de 2019
Atualização de versão da plataforma do Amazon EKS	Nova versão da plataforma que atualiza os clusters Kubernetes `1.11` do Amazon EKS para o nível de patch `1.11.8` para resolver `CVE-2019-1002100` .	8 de março de 2019

Aumento do limite de clusters	O Amazon EKS aumentou de 3 para 50 o número de clusters que você pode criar em uma Região da AWS.	13 de fevereiro de 2019
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível nas Regiões da AWS Europa (Londres) (`eu-west-2`), Europa (Paris) (`eu-west-3`) e Ásia-Pacífico (Mumbai) (`ap-south-1`).	13 de fevereiro de 2019
Nova AMI com patch otimizada para Amazon EKS para `ALAS-2019-1156`	O Amazon EKS atualizou a AMI otimizada do Amazon EKS para resolver a vulnerabilidade descrita em `ALAS-2019-1156`.	11 de fevereiro de 2019
Nova AMI com patch otimizada para Amazon EKS para `ALAS2-2019-1141`	O Amazon EKS atualizou a AMI otimizada do Amazon EKS para resolver os CVEs referenciados em `ALAS2-2019-1141`.	9 de janeiro de 2019
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível na Região da AWS Ásia-Pacífico (Seul) (`ap-northeast-2`).	9 de janeiro de 2019

Expansão da região do Amazon EKS	O Amazon EKS já está disponível nas seguintes Regiões da AWS adicionais: Europa (Frankfurt) (`eu-central-1`), Ásia-Pacífico (Tóquio) (`ap-northeast-1`), Ásia-Pacífico (Singapura) (`ap-southeast-1`) e Ásia-Pacífico (Sydney) (`ap-southeast-2`).	19 de dezembro de 2018
Atualizações do cluster do Amazon EKS	Adicionada documentação para atualizações de versão do Kubernetes para clusters e substituição de nós do Amazon EKS.	12 de dezembro de 2018
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível na Região da AWS Europa (Estocolmo) (`eu-north-1`).	11 de dezembro de 2018
Atualização de versão da plataforma do Amazon EKS	Nova versão de plataforma atualizando o Kubernetes para o nível de patch `1.10.11` para solucionar `CVE-2018-1002105`.	4 de dezembro de 2018
Adição de suporte para a versão `1.0.0` do controlador de ingresso do ALB	O controlador de ingresso do ALB lança a versão `1.0.0` com suporte formal da AWS.	20 de novembro de 2018

Adicionado suporte para a configuração de rede do CNI	O Amazon VPC CNI plugin for Kubernetes versão `1.2.1` agora é compatível com configuração de rede personalizada para interfaces de rede de Pod secundário.	16 de outubro de 2018
Adicionado suporte para `MutatingAdmissionWebhook` e `ValidatingAdmissionWebhook`	A versão `1.10-eks.2` da plataforma do Amazon EKS agora oferece suporte ao `MutatingAdmissionWebhook` e aos controladores de admissão `ValidatingAdmissionWebhook`.	10 de outubro de 2018
Adicionadas informações de AMIs de parceiros	A Canonical fez uma parceria com o Amazon EKS para criar AMIs de nós que você pode usar nos clusters.	3 de outubro de 2018
Instruções adicionadas para o comando `update-kubeconfig` da AWS CLI	O Amazon EKS adicionou `update-kubeconfig` à AWS CLI para simplificar o processo de criação de um arquivo `kubeconfig` para acessar o cluster.	21 de setembro de 2018
Novas AMIs otimizadas para o Amazon EKS	O Amazon EKS atualizou as AMIs otimizadas para o Amazon EKS (com e sem suporte a GPU) para fornecer várias correções de segurança e otimizações da AMI.	13 de setembro de 2018
Expansão da Região da AWS do Amazon EKS	O Amazon EKS já está disponível na região Europa (Irlanda) (`eu-west-1`).	5 de setembro de 2018

Atualização de versão da plataforma do Amazon EKS	Nova versão da plataforma com suporte a camada de agregação e ao Horizontal Pod Autoscaler (HPA) do Kubernetes.	31 de agosto de 2018
Novas AMIs otimizadas para o Amazon EKS e suporte a GPU	O Amazon EKS atualizou a AMI otimizada para o Amazon EKS, a fim de usar um novo modelo de nó do AWS CloudFormation e um script de bootstrap. Além disso, uma nova AMI otimizada para o Amazon EKS com suporte a GPU está disponível.	22 de agosto de 2018
Nova AMI com patch otimizada para Amazon EKS para `ALAS2-2018-1058`	O Amazon EKS atualizou a AMI otimizada do Amazon EKS para resolver os CVEs referenciados em `ALAS2-2018-1058` .	14 de agosto de 2018
Scripts de compilação de AMI otimizados para o Amazon EKS	O Amazon EKS usa scripts de compilação de código aberto para criar a AMI otimizada para o Amazon EKS. Esses scripts de compilação agora estão disponíveis no GitHub.	10 de julho de 2018
Lançamento do Amazon EKS	Documentação inicial para a inicialização do serviço	5 de junho de 2018